1987

Modeling and analysis of three-dimensional robotic palletizing systems for mixed carton sizes

Du-Ming Tsai
Iowa State University

Order Number 8805143

Modeling and analysis of three-dimensional robotic palletizing systems for mixed carton sizes

Tsai, Du-Ming, Ph.D.

Iowa State University, 1987

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark ✓.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy ✓
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Dissertation contains pages with print at a slant, filmed as received _____
16. Other _____

_____

_____

Modeling and analysis of three-dimensional robotic palletizing systems for mixed carton sizes

by

Du-Ming Tsai

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Major: Industrial Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1987

TABLE OF CONTENTS

I. STATEMENT OF PROBLEM

Material handling pallets are the most common tool used in warehousing industries. Nearly every warehouse uses them to some extent. Pallets have become an almost universal warehouse operations tool. They provide a convenient, simple way to transport, stack, and store materials.

Traditional palletizing methods load only boxes of the same size on one pallet. For retail business such as grocery distribution or manufacturers that produce many products of small quantities, a wide product mix of different box sizes must be loaded onto the same pallet. The traditional palletizing method may not optimize the utilization of the pallet cube.

Manual palletizing is an extremely tedious and fatiguing task. Automatic palletization is, therefore, a potentially attractive alternative. Commercially available palletizers handle only one box size at a time. They cannot meet the requirements of palletizing applications with mixed box sizes. Industrial robots have always been a viable solution to complex loading operations due to their flexibility and programming capability.

Conveyors are the most common device used to transfer boxes to the robotic palletizing station from warehouses or production lines. Since boxes of various sizes can randomly arrive via the conveyor, elaborate consideration must be given for the overall design of the robotic palletizing system. This enables the system to accommodate variations in the distributions of box sizes. Off-line storage areas may be required to absorb

boxes that cannot be immediately placed onto a pallet. These stored boxes can be picked up later when they can be successfully placed on the pallet.

This research is an extension of previous work by Tsai et al. [95, 96,97]. Their early efforts were concentrated on the development of two-dimensional pallet packing algorithm. Tsai's algorithm was static and did not respond to variations in the distributions of box sizes. In this research, three dimensional pallet loading with mixed box sizes has been investigated. This loading method allows many boxes of various sizes to be placed on the same pallet so as to maximize the pallet volume occupied by the boxes. The developed loading method is dynamically responsive to changes in the size distribution of the boxes in the loading queue. The completion of this research has involved the following tasks:

- Development of algorithms that specify an optimal three-dimensional pallet pattern for various mixes of box sizes when such boxes are loaded onto a pallet of fixed dimensions. The two-dimensional algorithm presented by Tsai is not directly extendable to three-dimensional pallet packing model. The developed three-dimensional algorithms are, therefore, new approaches.

- Development of a physical simulator of an integrated robotic palletizing system for both warehousing and manufacturing industries. The system design has focused on the problem of variations in box size distributions.

- Development of a robot control program for automatic palletization, and completion of a physical simulation of a robotic palletizing station. A Rhino XR-2 robot has been employed for this simulation. Data have been collected and analyzed during the simulation to evaluate the feasibility and performance of the robotic palletizing system.

A variety of literature has been written which addresses both robotic palletizing applications and mathematical algorithms of pallet packing problems. This literature is reviewed in the following chapter.

II. REVIEW OF RELEVANT LITERATURE

A. Introduction

A review of history reveals that robots are not new and that the application of robots to solve industrial problems dates back to the early 1960s. The use of robots in this capacity increased in the United States during the 1970s. Economic justification has delayed the widespread use of robots for a number of years. In 1960, the cost of operating a robot was over nine dollars per hour, while the overall cost of a human operator was less than five dollars per hour. When these figures are compared to equivalent data from 1982, the average labor cost has reached 15 dollars per hour. The robot operating cost has decreased to as low as 5 dollars an hour. The cost trends for both robots and labor are illustrated in Figure 2.1 [76].

Materials handling represents a large and growing part of the costs incurred by industry. All palletizing applications in materials handling are repetitive, multi-shift, highly labor intensive and tedious. These features provide a high possibility of economically justifying the use of robots for materials handling. As industry moves forward with automation and the utilization of industrial robots, robot palletizing applications will be increasingly attractive.

When compared with manual loading, robot palletizing has the following advantages [88].

- Increased productivity. Compared with the manual operations, the productivity can be increased by more than a factor of two.

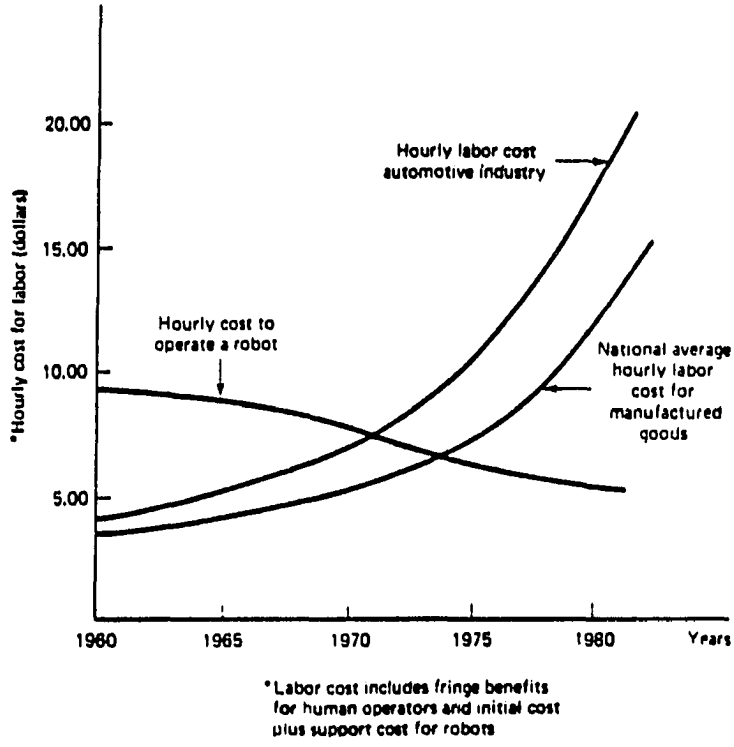


Figure 2.1. Robot and human labor cost (From reference [76])

- Stability and improvement in product quality. Smooth robotic movements and high positioning accuracy can reduce product damage during palletizing.
- Prevention of labor accidents and improvement of safety. Workers can be released from the task of positioning heavy boxes by hand as well as from an undesirable working environment.
- Savings in labor. A labor savings of one worker/shift can be achieved.
- Improved labor turnover. Because of the monotonous, heavy labor, the labor turnover can be reduced.

- Improvement in production management. Cycle time variation is reduced. The production schedule can be worked out based on the cycle time of the machine, making the production management easier.

B. Robotic Palletization

Robot palletizing offers advantages in automating materials handling operations using existing technology. Robotic palletization has been studied by many researchers in recent years. Their palletizing systems are designed only for the packing of identical boxes. A survey of robot applications for palletizing is presented in the paragraphs that follow.

1. Representative palletizing systems for industrial boxes

Abair [1] has investigated the possibility of using industrial robots to solve traditional palletizing problems. System layout, tooling, robot control software, interfacing, safety and possible manual operation are considered. He states that production rate requirement is the most critical factor in palletizing applications. A typical robot cycle is approximately 10-15 seconds from piece-part pickup to pickup. He suggests that robots may provide a cost effective solution if short-run products of different sizes with different palletizing patterns are produced. In terms of robot mechanical considerations, cylindrical and Cartesian coordinate styles are utilized more often than spherical and articulated arm styles according to the author.

Maximizing overall operations and ensuring system continuity are the main concerns of the design of the total system layout. Abair con-

siders the total floor space requirements with regard to robot work envelope, support item storage, input area, output area, maintainability and possible manual operation in case of machine breakdown. A possible palletizing layout is shown in Figure 2.2.

A robot palletizing center (RPC) has been developed by Grab [45]. The "factory of the future" is introduced as a complete system solution by an integrated production-logistics system called "INPROLOG" (see Figure 2.3). This system consists of a robot palletizing center (RPC), automatic guided vehicle systems (AGVS), an automatic forklift system (AFS) and collecting robots (COROB).

The author also considers the selection between a flexible robot palletizing center and single purpose unit. An application example in a dairy factory has been cited. The system layout is shown in Figure 2.4. The robot receives empty containers on "A", and feeds them to an input conveyor "C". They are next transferred to output conveyor "D". Here, they are retrieved by the robot and palletized on "B". At station "A", a container positioning stop gate is installed for alignment of boxes.

2. Palletizing cell characteristics

Modern Materials Handling [75] has reported that because of the low speed of robots, they will not displace high-speed palletizers for case packing. Six transfers per minute is reported as a required cycle time. This is slow when compared with special palletizer machines that can handle 20 to 100 or more identical cases per minute. Because of robots' flexibility of control and movement, they are beginning to be employed

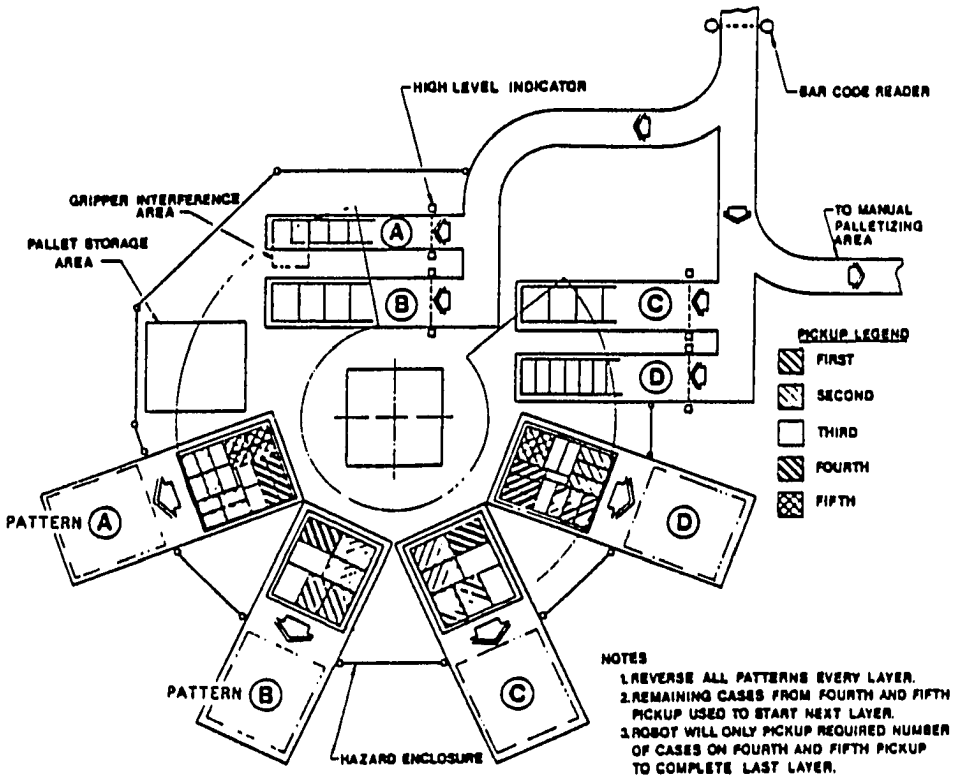


Figure 2.2. Palletizing work station
(From reference [1])

in low-speed, complex loading operations.

To increase packing speed, use of a vacuum gripper is recommended to pick up multiple cases simultaneously. However, the loading is limited to case weights of about 50 lbs. A combination mechanical-vacuum

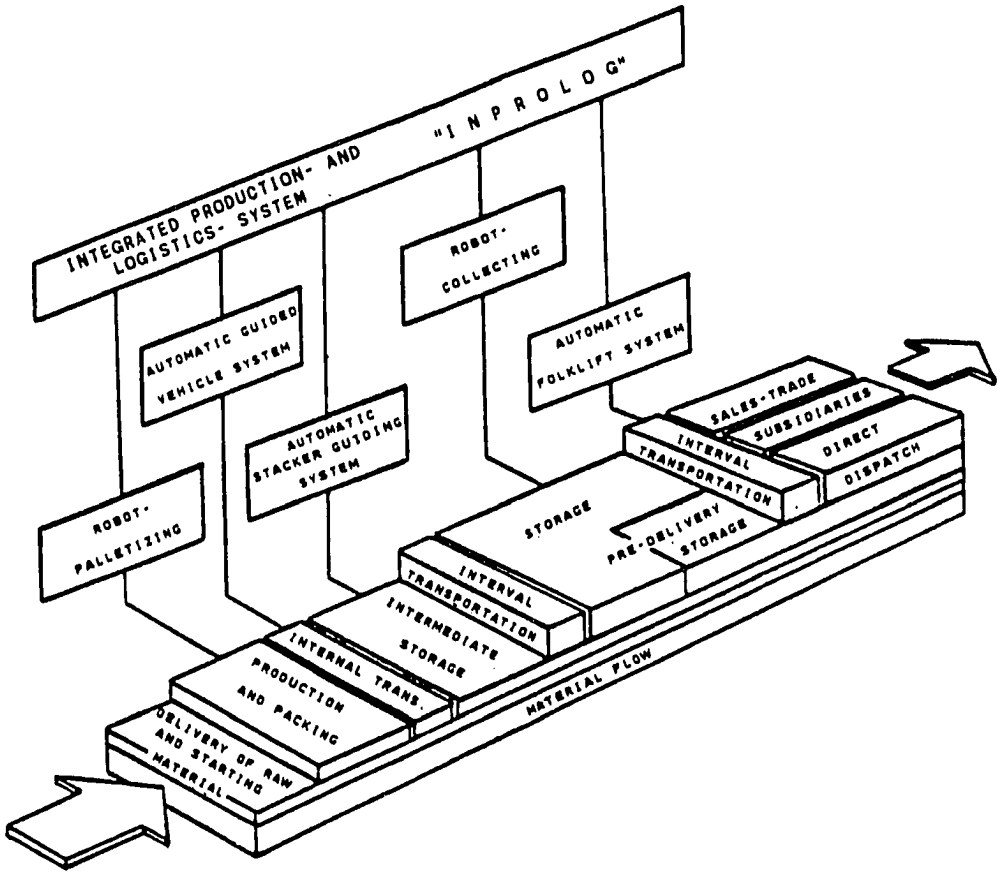


Figure 2.3. Integrated production and logistics system (From reference [45])

gripper is then recommended. A mechanical gripper can grasp and position a pallet. A vacuum gripper then picks up and transfers cases.

The installation of palletization station which uses up to eight pallets at Texas Color, Inc. is shown in Figure 2.5 [75]. The company uses a hydraulic cylindrical coordinate robot. Upon receiving a signal from the conveyor, the robot retrieves a bundle from the conveyor pick-up station. After a fork truck removes full loads, the robot positions

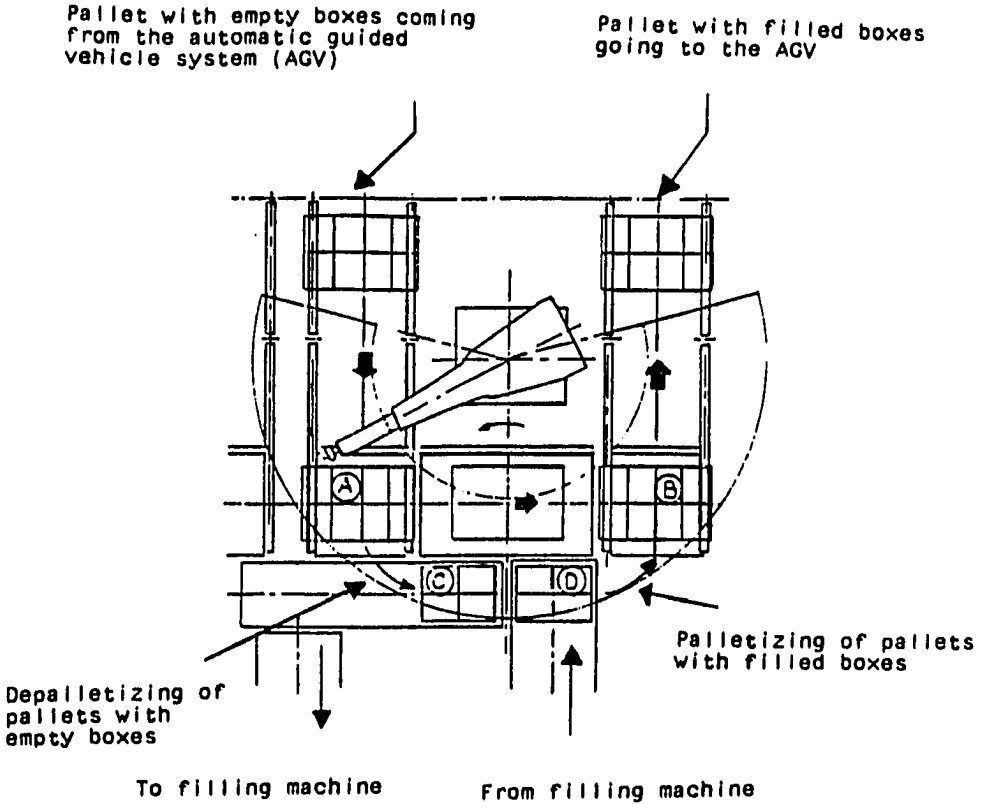


Figure 2.4. Robot palletizing station
(From reference [45])

an empty pallet. A special gripper, which retracts when not in use, allows the robot to grasp the pallets.

Modern Materials Handling [100] has also suggested that a robotic palletizer may be selected if lightweight cases of 40 lbs or less are handled, and the resultant packing rate ranges from 5 to 15 cases per minute.

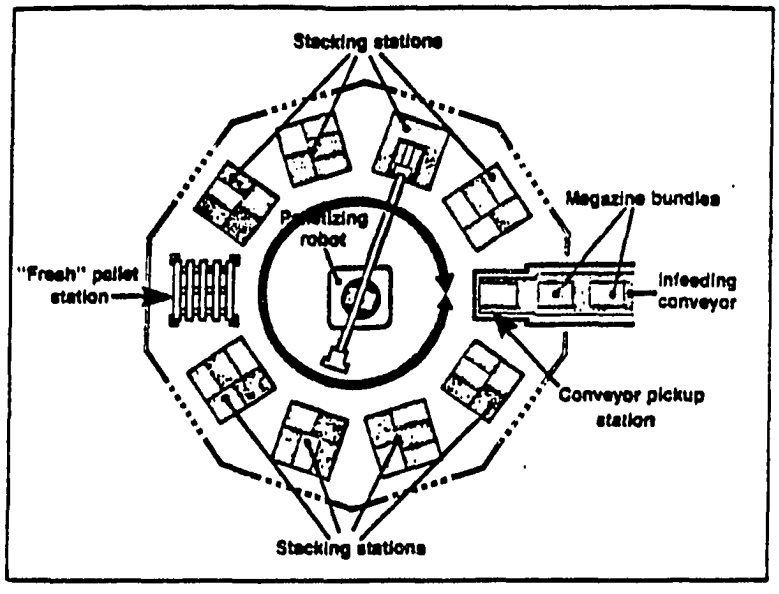


Figure 2.5. Palletizing station with eight pallets (from reference [75])

3. Sorting/stamping applications and multiple outputs

Schiwarov and Yanakiev [84] have examined a mechanical handling system designed for automatic separation and quality grading of wall tile packages in a standard ceramic factory. A robot is used for the stacking of the quality identified packages in stable pallet units. A full pallet is then automatically removed outside the palletization zone to the warehouse. A robotized mechanical handling system has been developed. The system layout is illustrated in Figure 2.6. The system consists of the following components (the number corresponds to the device number shown in Figure 2.6):

1. An industrial robot
2. A conveyor for carrying and automatic quality grading of wall tile packages
3. Mobile transport platforms
4. A spare gravitational roller conveyor for taking up unrecognized packages
5. A quality identification package system
6. A device for individual package release
7. Cycle mobile stops for stopping or releasing the packages

A Transman 2000 robot has also been used for palletizing [21]. The system layout is shown in Figure 2.7. The robot can travel along the base track so that two or more pallets can be packed at the same time.

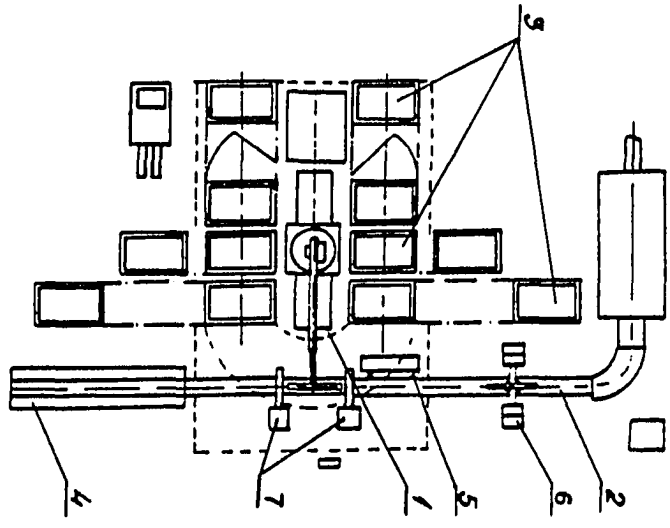


Figure 2.6. A robotized handling system (from reference [84])

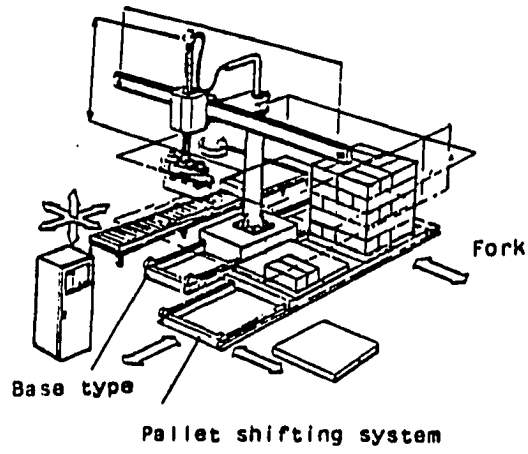


Figure 2.7. Transman 2000 robot for palletizing (from reference (71])

Material Handling Engineering [80] has discussed the possibility of using the robot as a palletizer. Palletizing robots can be integrated into the material handling systems, which can take directions and change the control program using information from photo-electric controls or scanners. Robots also handle multiple outputs from a material handling system. They can travel across the face of a number of terminating production lines to handle each product. Cycle time, production rate, case weights and physical layout are considered when automating this loading process.

Modern Materials Handling [78] has cited a robotic palletizing application for a stamping press factory. A conveyor, a robot, and a press work together to form an automated work cell. When a pallet is full, a light above a nearby post flashes to notify a fork truck operator.

It is then shuttled to an unload position, and an empty pallet automatically advances into place. Counting of workpieces is also accomplished by the robot's control program.

4. Palletizing cells with sensory features

Cotter and Batchelor [26] have concentrated on the visual monitoring of palletizing and packing. The sensing medium is used to identify the carton type. It also checks for correct location and orientation, ensures good stacking, and identifies full/empty locations on a partly filled pallet. Regions of constant height can easily be identified by the developed sensing system. These can be analyzed further for position/orientation information, e.g., the directions of moments of least insertia. This helps ensure the construction of stable unit loads.

Goto and Takeyasu [44] have studied a robot equipped with a computerized tactile member that can feel the forms of various objects and their positions, and load them compactly into a pallet. The robot was employed to feel for objects, and recognize the form and the position of an object. This information is used to select the best methods of positioning the object on a pallet.

All robotic palletizing systems reviewed in this section consider only one box size at a time. Each palletizing procedure is relatively simple and straightforward when compared with that of mixed box sizes. Off-line box storage requirements and box distributions coming off a conveyor are never the problems for the palletization of identical box sizes. However, elaborate design is required for the palletization of

mixed box sizes to overcome these problems. Off-line box storage and box distributions will be addressed in a later chapter.

C. Pallet Loading Problems

1. Introduction

The problem of nesting varied shapes into fixed rectangular dimensions has long been a challenge for researchers. The problem has been called tessellation, plane tiling or plane paving. It was first brought into mathematical prominence by Hilbert and Vortrag [54] in 1900.

The pallet loading problem is generally addressed by attempting to maximize the number of small rectangles that can be placed orthogonally within a large containing rectangle. This problem was also discussed in relation to an expanse of operations research literature on the two-dimensional cutting stock and bin packing problems. Cutting stock has been the problem of determining and cutting a set of stock sheets to satisfy a given order. In bin packing problems, the objective is normally to pack a given set of rectangular pieces according to some simple placement rules. The pieces are loaded into a bin of fixed width and infinite height, so as to minimize the height required.

The pallet loading problem falls into a class of problem termed NP-Hard [37]. No polynomial algorithms exist for this class, and consequently, it is intractable. A problem is defined to be intractable if all algorithms to solve that problem are of at least exponential time complexity. Therefore, a truly efficient optimal algorithm is not like-

ly to be forthcoming. The combinatorial nature of the problems has often led to the use of heuristic methods. The heuristics applied are closely linked to the characteristics of the particular problem being addressed.

Of importance is the work of Gilmore and Gomory [39,40,41] in their development of knapsack functions for the cutting stock problem in the early 1960s. Their algorithm is based on linear programming and dynamic programming techniques. This approach has inspired a number of authors for the past two decades.

2. Overview of literature

A review of the literature in the area of loading and cutting problems has been reported by many authors. Tsai [95] has classified this type of problem into plane tiling, pallet stacking, stock cutting, bin packing and container packing. His survey was based on articles published from 1960 to 1982. These articles are briefly summarized in the paragraphs that follow.

The plane tiling problems seeks to pave a finite number of square or rectangular hexagons, without gaps or overlaps, to comprise a large plane figure of specified dimensions. This problem has been studied by Basin [8], Chung et al. [22], Kershner [62], Graham [46], and Hoffman [59].

For pallet stacking, researchers have concentrated on the investigation of fitting a number of identical boxes into a pallet that will maximize the usage of pallet space. Papers reviewed in this area in-

clude Steudel [90,91], Tanchoco and Agee [93], Salzer [82], Kulick [64], Gupta [47], and Smith and de Cani [87].

Literature relating to one- and two-dimensional stock cutting problems in Tsai's survey include Gilmore and Gomory [39,40,41], Haessler [49], Herz [53], Hahn [50], Albano and Orsini [4], Adamowicz and Albano [2,3], Albano and Sapuppo [5], Cheng and Pila [19], Page [74], Haims and Freeman [51], and Christofides and Whitlock [20]. Integer programming, dynamic programming and tree search algorithms are the most frequent methods used to approach the stock cutting problems.

Bin packing problems have been the subject of a number of papers including those of Chung et al. [21], Baker et al. [6], and Brown [16].

George and Robinson [38] are the only authors who directly address the three-dimensional problem in Tsai's survey. The authors studied the problem of finding a way to pack a shipment of boxes of various rectangular shapes into a shipping container.

A more detailed summary of these investigations has been previously presented by Tsai [95]. Interested readers are urged to refer either to this source or to references [2,3,4,16,20,38,39,40,41,47,53, and 91].

An outstanding survey has also been presented by Dowsland [30]. This article reviews work published up to 1984 which is of direct relevance to the solution of two- and three-dimensional packing problems. This particular problem has also been the subject to a number of survey papers including those of Brown [15], Golden [43], Hinxman [56] and Israni and Sanders [60].

Garey and Johnson [36] have concentrated on the survey of 1-dimensional and 2-dimensional bin packing problems. Instead of using sophisticated mathematical programming techniques, bin-packing algorithms usually consist of a specified ordering of pieces, and very simple placement rules. Worst case analysis is usually carried out for evaluating the performance of the rules. In this excellent survey, 13 one-dimensional and more than 10 two-dimensional bin-packing rules are reviewed.

The survey in the following sections is based on the articles published from 1980 to 1985 which contribute directly to the problems of two-dimensional stock cutting, pallet loading and bin packing. Important theorems and procedures that would apply directly or indirectly to implement the developed algorithm of this research will be described in detail.

3. Linear and goal programming approaches

Tsai [95] and Tsai, Malstrom, and Meeks [96,97] have formulated a linear programming (LP) model to solve the pallet loading problem. The box sizes are restricted to the same heights in their study. Since the box heights are identical, the solution is two-dimensional. Nevertheless, the number of box sizes, box lengths and box widths can be selected arbitrarily.

For an ordered pair (l_i, w_i) or L x W, define the first element to be the length, which is referred to as a rectangle's horizontal edge. Likewise, the second element represents the width, which is the rec-

tangle's vertical edge. (l_i,w_i) denotes the dimensions of a small piece i. (l_i,w_i) and (w_i,l_i) are considered to be two different box types. L x W denotes the dimensions of a pallet.

The first step of Tsai's procedure is to horizontally divide the pallet to W strips, each having the length L and width 1. These further reduce the two-dimensional loading problem to the one-dimensional one. If a combination of small pieces can fully cover the area of a pallet, each strip must be a multiset that contains the lengths of the small pieces that the strip intersects, and have a total sum of L. By enumerating all possible combinations of strips, the relationships between the pieces' lengths and the pallet's length are defined.

For example, consider three types of small pieces with sizes (2,2), (3,4) and (4,3), and a pallet of sizes 8 x 7. The only possible strips that have the sum of 8 are:

{4,4},

{2,2,4},

{2,3,3},

{2,2,2,2}.

The second step is then to set up the constraints of widths by considering the strip types and pieces' widths. The LP constraints are formulated as follows:

Let y_j = decision variable, the number of type j strips,

$j = 1,2,\ldots,m$

x_i = decision variable, the number of type i pieces,

$i = 1,2,\ldots,4$

a_{ij} = number of length i in strip j

r = total number of box types to be considered.

The first constraint is to limit the total number of strips to the width of the pallet. This gives

$$\sum_{j=1}^{m} y_j = W \tag{2.1}$$

Then, for each different piece's length, the number of times that this length is used in all m strips must equal to the sum of all of the widths of the r boxes that have this specific length. Thus, for a given length k,

$$\sum_{j=1}^{m} a_{kj} y_j = \sum_{i=1}^{r} \delta_{ki} w_i x_i \tag{2.2}$$

where δ_{ki} = 1, if l_i = length k

= 0, otherwise

There is no guarantee that a combination of small pieces always exists that can completely fill the pallet. Therefore, a dummy square piece with the length of one is introduced to ensure that there is at least one combination of pieces that can fully cover the pallet's area. The number of dummy unit pieces represents the wasted area of the pallet. The upper bound of the number of the dummy pieces is required so that the number of possible strip types can be minimized. Recall that each strip type represents a decision variable in the LP model.

A simple pallet pattern is to place identical pieces on the same pallet. The upper bound of dummy square pieces is determined based on the principle that if the optimal solution obtained by using the formulated LP model is no better than that of the simple pallet pattern, then the simple pallet pattern should be employed for optimal packing. The algorithms of Brualdi and Foregger [17], and Barnes [7], which will be described in detail in a later section, are excellent tools to determine the upper bound of the number of dummy unit pieces.

A complete LP model for the two-dimensional pallet packing problem thus has the form:

$$\text{maximize } Z = \sum_{i=1}^{r} c_i x_i \tag{2.3}$$

subject to

$$\sum_{j=1}^{m} y_j = W \tag{2.4}$$

$$\sum_{j=1}^{m} a_{kj} y_j = \sum_{i=1}^{r+1} \delta_{ki} w_i x_i \text{ , for all different lengths } k \tag{2.5}$$

where $c_i = l_i \cdot w_i$

x_{r+1} = the number of dummy square pieces.

This formulated LP model can be easily solved using existing mathematical software systems. It is efficient when compared with the existing algorithms reviewed in Tsai's survey. However, the optimal pallet

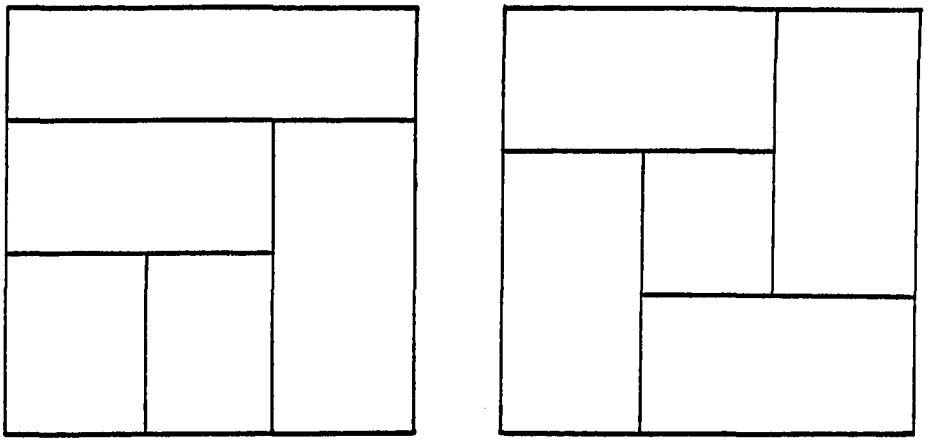
pattern is generated implicitly from the LP model. Only the number of pieces for each type, orientations of pieces, and the number of strips for each type are given from the final solution instead of the location of each piece on a pallet. Human intervention may be required to determine the solution's corresponding pallet pattern. Tsai et al. [95,96,97] constructed a physical simulator to evaluate different loading methods using the developed LP model.

Fleming [33] and Fleming, Malstrom and Meeks [34] have extended the early work of Tsai. Their developed model is still two-dimensional but is responsive to changing distributions of box sizes that arrive at the robot to be palletized.

Two goals must be achieved in the model. First, each pallet has to accommodate a proportional number of boxes of each size so that no size of box is left unpalletized. Second, the pallet has to be as full as possible. Linear goal programming technique is employed to approach the formulated multi-objective model.

4. Dynamic programming approaches

Beasley [9] has studied the two-dimensional, guillotine cutting problem of cutting a single large rectangular plane to a number of small rectangular pieces. His objective is to maximize the value of the pieces cut. A guillotine cut on a rectangle is a cut from one edge of the rectangle to the opposite edge which is parallel to the two remaining edges. Figures 2.8a and 2.8b show the guillotine and nonguillotine cutting patterns.



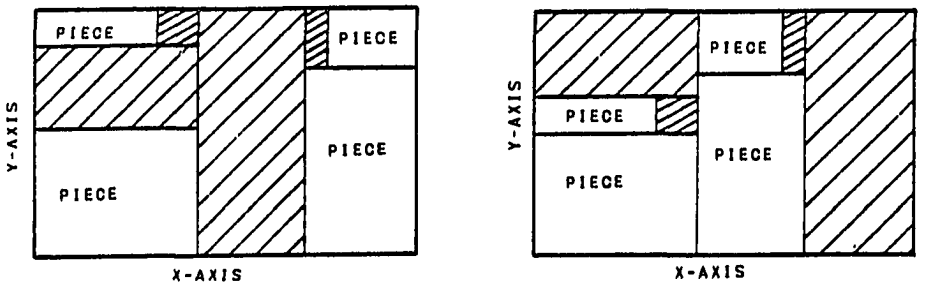
(a) guillotine pattern (b) nonguillotine pattern

Figure 2.8. Two cutting patterns

There are no constraints upon the number of pieces of each type that are cut from the large plane. The author points out the error of the recursion procedure given in Gilmore and Gomory [41], and presents a method of dynamic programming recursion for staged cutting which develops different functions for different first-stage cut directions. A heuristic algorithm is also presented when the optimal recursive procedure becomes computationally infeasible.

The dynamic programming recursion for staged cutting is enhanced computationally through the use of normal patterns. Figure 2.9 shows a nonnormalized cutting pattern and a normalized cutting pattern. Christofides and Whitlock [20] have proved that given a cutting pattern, any piece/cut in that pattern can be "left-shifted" until both the left-hand edge and the bottom edge of pieces are adjacent to a cut. The

normalization preserves any stage property in the original cutting pattern.



(a) nonnormalized (b) normalized

Figure 2.9. Nonnormalized and normalized pattern (from reference [9])

Steudel [92] has extended the 1979 recursive algorithm of Steudel [91] to a new heuristic and provided more efficient loading patterns. The algorithm is limited on two-dimensional nonguillotine cutting fashion, and only identical size rectangular items are considered.

The heuristic model consists of three phases. Phase one utilizes a dynamic programming recursion to determine four optimum sets of length and/or width placements of the item along the inside edges of the pallet. Phase two utilizes heuristic rules to project the optimum perimeter arrangement inward to fill the center of the pallet. Phase three first attempts to explode the phase two pallet pattern to the perimeter edges of

the pallet. It next inserts end and/or side-stacked items to maximize the number of items per layer. It can be shown that side stacking of boxes, when applicable, can yield average increase in the range of 5% in number of items per pallet load.

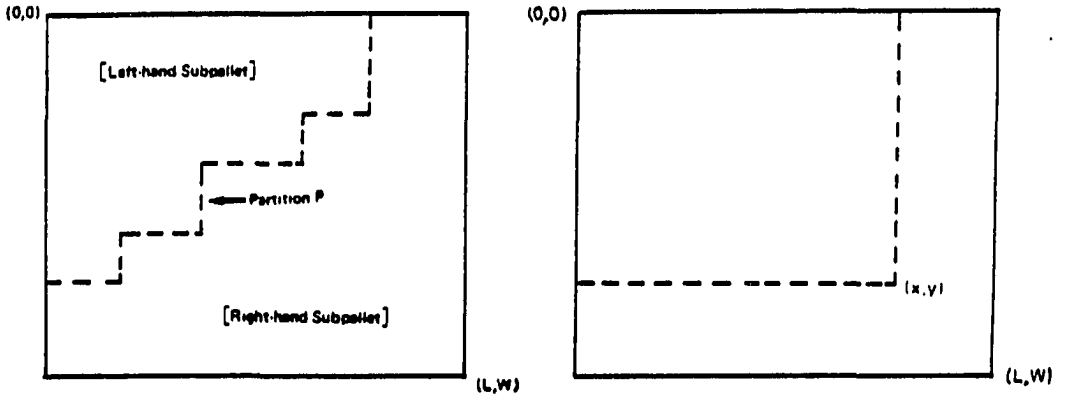
Hodgson [57,58] has addressed the problem of two-dimensional pallet loading with a set of various sizes of rectangular boxes. The problem is approached using a combination of dynamic programming and heuristics. In this dynamic programming model, a partition divides the pallet into two parts of echelon-shape. The partition values in each stage are then calculated. Since the number of possible partitions of the pallet is extremely large, partitions are limited to rectangles to reduce computer storage and computation time. Figures 2.10a and 2.10b show the echelon-shaped and rectangular partitions.

The structured solutions resulting from the dynamic programming model allow any item to be placed on the periphery of the pallet for easy access. Also, some control may be retained over the pallet's center of gravity.

5. Heuristic approaches

To reduce computation effort, heuristics have been introduced to solve the pallet loading problem. In all cases, the number of potential configurations is reduced by considering only layouts of a particular form.

Beasley [10] has investigated the two-dimensional, guillotine cutting assortment problem where constraints are imposed upon the number of small pieces of each type. A heuristic algorithm is developed based on a pro-



(a) echelon-partition (b) rectangle-partition

Figure 2.10. Pallets with partitions (from reference [57])

cedure for generating two-dimensional cutting patterns. A linear program for choosing the cutting patterns is used. Also, an interchange procedure determines the best subset of stock rectangles to cut.

Farley [31,32] has studied the two-dimensional, guillotine cutting, trim-loss problem arising in the glass-industry. A restriction that imposes a limitation upon the positioning of cut within the stock-plate is added to the conventional guillotine cutting problem. Two procedures are presented which take an existing pattern and attempt to rearrange it to meet the restrictions. First, the heuristic procedure attempts to transpose one of the cuts from a pair of cuts not currently complying with the restrictions. Secondly, the enumeration procedure generates all possible patterns corresponding to the required combination. The

patterns must conform to the data structure chosen that represents a cutting pattern.

Roberts [77] has addressed the cutting-stock problem which arises in the manufacture of furniture. The worktops to be cut are of one of two basic shapes, either rectangular or L-shaped. Due to the relative dimensions of a typical worktop and the raw material used, the approach to a solution has been to reduce the problem to a series of one-dimensional trim-loss exercises. A heuristic was developed to schedule the cutting of worktops of varying shapes and sizes from available raw material. The method of solution was to order the raw material in order of increasing size. Each stage of the solution process involves an attempt to remove the largest remaining worksop from what remains of the current piece of raw material. Based on numerical experiments, it was shown that the overall waste incurred for the complete set of runs involving 1491 worktops was 12.6%.

Israni and Sanders [60] have investigated a two-dimensional cutting stocking problem. A recursive algorithm to lay out rectangular bills of material on stock sheets is presented. This algorithm is based on the First-Fit Decreasing-Height[1] (FFDH) algorithm used in bin packing problems. The procedure allows human intervention at strategy points during its run. In the process, the human may choose to fill in the resultant gap of the stock sheet with any judicious combination of unallocated pieces. Intervention continues until the user is satisfied.

---

[1]The FFDH algorithm will be discussed in a later section.

6. Integer programming approach

Beasley [11] has considered the two-dimensional, nonguillotine cutting problem of cutting a number of rectangular pieces from a single large rectangle so as to maximize the value of the pieces cut. An exact algorithm based on a zero-one integer programming was developed. Beasley states that no other exact solution procedure for this problem exists in the literature.

The formulation is a large zero-one integer programming involving (MxLxW) variables and (M+LxW) constraints. Here M is the total number of types of small pieces. L and W are the length and width of the large stock sheet. The Lagrangian relaxation is then used as a bound in a tree search procedure. Subgradient optimization is also employed to optimize the bound derived from the Lagrangian relaxation.

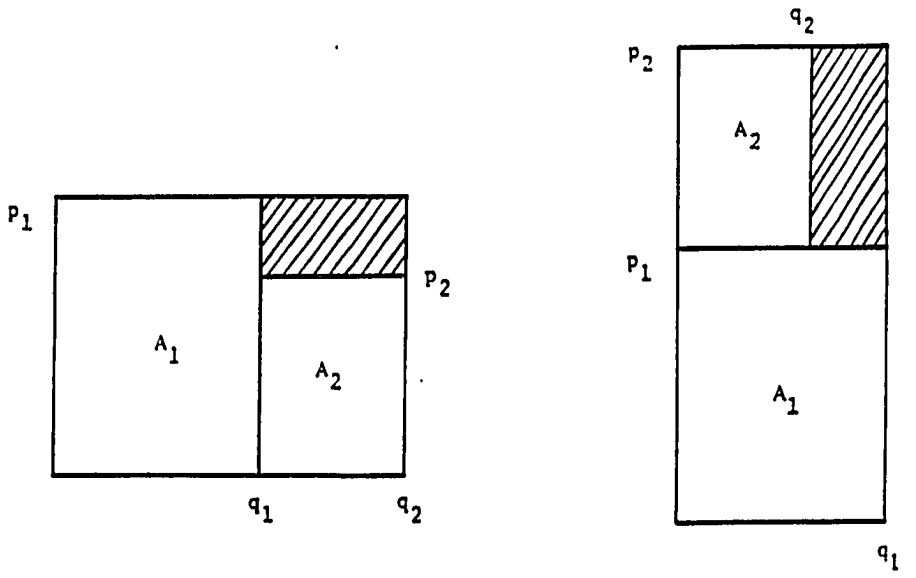
Besides Lagrangian relaxation, Beasley [12] has also proposed linear programming relaxation, and knapsack relaxation to set up the upper bound for the tree search procedures. By releasing the binary variables, x, to the constraint of $x \geq 0$, this linear programming relaxation becomes a large L.P. model. The relaxed program using knapsack relaxation is the problem of filling a knapsack of size L-by-W (the large rectangle) with a number of items of size l-by-w (the small pieces) so as to maximize the value of the items in the knapsack. This problem can be solved by dynamic programming for the upper bound value.

7. Combinatoric and network flow approaches

Wang [99] has proposed two combinatoric methods that generate constrained two-dimensional, guillotine cutting pattern by successive horizontal and vertical builds of ordered rectangles. A horizontal build of two rectangles $A_1 = p_1 \cdot q_1$ and $A_2 = p_2 \cdot q_2$ is a rectangle having dimensions $\max(p_1,p_2)$-by-(q_1+q_2). A vertical build of A_1 and A_2 is a rectangle of dimensions (p_1+p_2)-by-$\max(q_1,q_2)$. Figures 2.11a and 2.11b illustrate a horizontal and vertical build of A_1 and A_2.

Instead of enumerating all possible cuts, the combinatoric algorithm finds guillotine cutting patterns by successively adding the rectangles to each other. Two parameters having value between 0 and 1 are employed to reject undesirable additions. The first algorithm measures the parameter with respect to the area of large stock sheet. The second algorithm measures the other parameter with respect to the area of guillotine rectangle generated in the algorithm. Theorems are also developed for determining the error bounds that measure the closeness of the best patterns to the optimal solution.

Biró and Boros [13] have presented a network flow algorithm to approach the nonguillotine cutting problem. The cutting problem is interpreted in a resource contained scheduling context. In this case, W (width of a large stock sheet) denotes the amount of resource available at any time. The width of a small piece i, w_i, denotes the amount of resource required by job i at all times during its execution. The length of a small piece i, l_i, denotes the processing time of job i. The ob-



(a) horizontal build (b) vertical build

Figure 2.11. Builds of rectangles A_1 and A_2 (from reference [99])

jective is to minimize the maximum completion time without preemption, i.e., minimize the total length used of the large stock sheet. Theorems are developed that contain the algorithms which are able to construct a cutting pattern from a flow with the required properties. The characterization makes it possible to search for good cutting patterns with the help of network flow and graph theory techniques. Here, an arc with positive flow represents the direct superposition of two rectangles. The capacities of the arcs are the maximum sizes of the intervals along which they can be superposed.

8. The tiling aspect

Two important theorems have been developed for two-dimensional tiling with identical bricks. One is for harmonic bricks. The other

is for the bricks that have relatively prime dimensions.

Brualdi and Foregger [17] have studied the problem of finding the largest number of identical bricks that can be packed in the box with the sides of the bricks parallel to the sides of the box. Here, the brick sizes are restricted to be harmonic. An a x b brick (where $b \geq a$) is harmonic if b is a multiple of a. Let [t] denote the largest integer value no greater than t. The authors have proved the following theorem:

The maximum number of harmonic bricks of fixed size a x b that can be packed in a L x W box is

$$N(L,W;a,b) = [L/b][W/b](b/a) + [W/b][m/a] + [L/b][n/a] + \max(0, [m/a] + [n/a] - b/a) \quad (2.6)$$

where N(L,W;a,b) = maximum number of bricks,

$m = L - [L/b](b)$,

$n = W - [W/b](b)$.

Barnes [7] has also addressed the problem of packing the maximum number of a x b tiles in a large L x W rectangle. Here, a and b are relatively prime integers. Let A be the waste in an optimal packing of a x 1 tiles on the L x W rectangle, and B the amount of waste in an optimal packing of b x 1 tiles. The value of A is given by

$$A = \min\{rs, (a-r)(a-s)\} \quad (2.7)$$

where r = L(mod a);

s = W(mod a);

mod = the modulus division, which gives the remainder when the first argument is divided by the second.

B is obtained by a similar calculation. Define R to be the least non-negative integer such that

$$R \geq \max(A, B), \text{ and} \tag{2.8}$$

$$R \equiv A(\text{mod } a);$$

$$R \equiv B(\text{mod } b).$$

The author has proved that if L and W are sufficiently large, then the wasted area in the best possible packing of a L x W rectangle with a x b tiles is precisely R.

Dowsland [28] has pointed out that "sufficiently large" requirement of Barnes' theorem implies that the ratio of the containing rectangle to the contained rectangle is larger than that encountered in most pallet loading problems. However, Barnes' estimate can be employed as an upper bound on the optimal packing number. The author then modifies Barnes' procedure and introduces a partition reduction technique to approach the packing problem.

9. The loading aspect

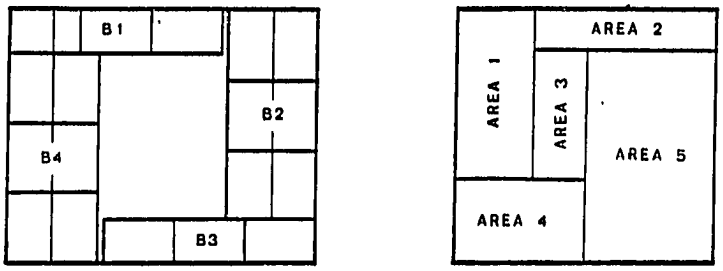
The two-dimensional orthogonal packing problem of packing identical rectangles into a large containing rectangle has been studied by Dowsland [29]. The following theorem is proved. The set of feasible layout orthogonal packings of identical rectangles into a large rectangle

is totally defined by the set of efficient partitions. Here, an efficient partition of a pallet edge S, in box dimensions a x b, is an ordered pair of nonnegative integers (n,m) satisfying

$$0 \leq S - (na+mb) < b. \tag{2.9}$$

The author then examines the condition under which the set of feasible layouts remains unchanged and shows that these conditions can be represented by a series of planes in three-dimensional space. A graphic representation of the relationship between box and pallet dimensions in three-dimensional space yields a method for obtaining accurate outlines for standard two-dimensional pallet charts. Thus, a three-dimensional pallet chart is constructed. Such charts can be used to measure the sensitivity of a layout to change in box and pallet dimensions. The results of the theorem are also used to provide an estimate of the optimal packing number of small rectangles.

Bischoff and Dowsland [14] have illustrated an application of the micro-computer to the problem of pallet loading with identical cases. Their algorithm is based on the early developments of Steudel [91] and Smith and de Cani [87] who divide the pallet into up to four rectangular areas. The investigation of the shortcomings of Steudel's approach and the method of Smith and de Cani leads to an improved algorithm for the loading rectangular pallets that allows for layouts with up to five blocks. The pallet layouts of four and five areas are illustrated in Figure 2.12.



(a) Four-area method (b) five-area method

Figure 2.12. Pallet layouts with blocks (from reference [14])

Carpenter and Dowsland [18] have given practical consideration to the development of loading patterns into pallet stacks. Criteria are also presented to determine the suitability of the pallet stacks for storage and transportation. Three criteria for stability requirements are:

- Each box must have its base in contact with at least two boxes in the layer below. Contacts of less than X% of a box's base area will be ignored.
- Each box must have at least Y% of its base area in contact with the layer below.
- There must be no straight or jagged guillotine cuts traversing more than Z% of the stack's maximum length or width.

10. The bin packing aspect

Bin packing algorithms usually consist of a specified ordering of pieces and a placement policy. The analysis applied to the performance

of the algorithms is in the form of worst-case analysis in which the ratio of the actual solution to the optimal solution is examined.

Coffman et al. [23] have analyzed three level-oriented algorithms for packing rectangles into a unit-width, infinite-height bin so as to minimize the total height of the packing. The three algorithms they discuss are summarized below:

- Next-Fit Decreasing-Height (NFDH): Let L be an arbitrary list of small rectangles. L is ordered by nonincreasing height. Also, a level is defined by a horizontal line drawn through the top of the maximum height rectangle placed on the previous level. The first level is simply the bottom of the bin. With the NFDH algorithm, rectangles are packed left-justified on a level until there is insufficient space at the right to accommodate the next rectangle.

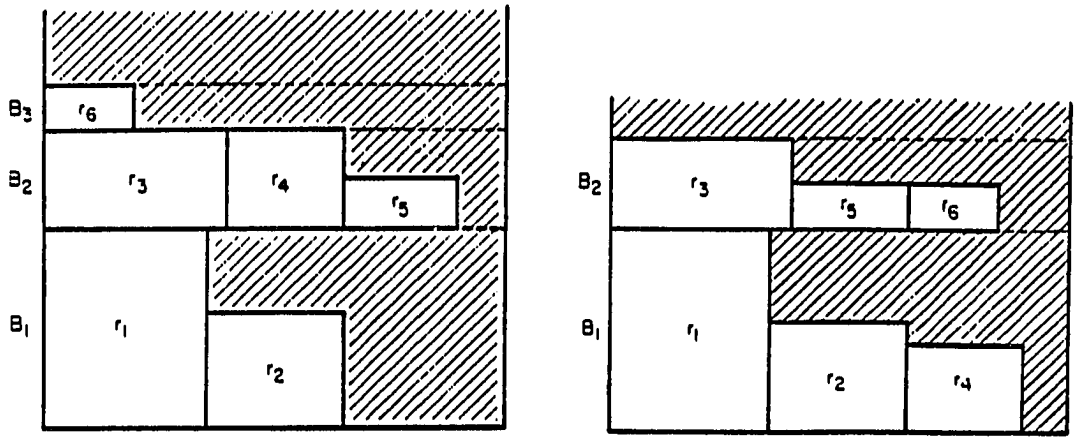
- First-Fit Decreasing-Height (FFDH): Let L be ordered by nonincreasing height. In the packing sequence, the next rectangle to be packed is placed left-justified on the first level on which it will fit. If none of the current levels will accommodate the rectangle, a new level is started as in the NFDH algorithm.

- Split-Fit algorithm (SF): Divide the given list L of rectangles into two sublists L1 and L2 according to pieces' widths. Then, pack the rectangles in L1 using the FFDH algorithm. Rearrange the block of this packing so that sufficient room is created. Finally, pack the rectangles in L2 into the room established by L1. The remainder will be packed above the packing for L1 using the FFDH algorithm.

Figures 2.13a and 2.13b demonstrate the packings of six small rectangular pieces using the NFDH and FFDH algorithms.

The authors prove that for these three algorithms, the ratio of the height obtained by the algorithm to the optimal height is asymptotically bounded by 2, 1.7 and 1.5, respectively.

Two orthogonal oriented two-dimensional, packing algorithms are



(a) NFDH algorithm (b) FFHD algorithm

Figure 2.13. Packings of six small rectangles (from reference [23])

also presented by Golan [42]. One is called Split algorithm (SP-algorithm). The other is called Mixed-algorithm (M-algorithm). The SP-algorithm splits an open-ended rectangle into two open ended rectangles and a closed rectangle. The M-algorithm divides the pieces into several sets and packs the pieces in the set in different ways. The author shows that for the M-algorithm the ratio of the height used by the algorithm to the optimal height is asymptotically bounded by 4/3.

Baker et al. [6] have developed an algorithm called the Up-Down (UD) algorithm for a two-dimensional bin packing problem. The UD algorithm is derived by analyzing the behavior of the Next-Fit Decreasing-Height

(NFDH) and First-Fit Decreasing-Height (FFDH) algorithms. The author proves that the ratio of the height obtained by the new UD algorithm to the height used by an optimal packing is asymptotically bounded by 5/4. This bound is the best value that has been proved so far in the bin packing literature.

All algorithms reviewed in this chapter are limited to two-dimensional problems. Two new mathematical algorithms for the three-dimensional pallet packing problem and robotic palletizing systems for mixed box sizes are the subjects of the following chapters.

III. THREE-DIMENSIONAL PALLET LOADING ALGORITHMS

A. Introduction

There are two major facets of the pallet packing problem [57]. The first is called the manufacturer's pallet packing problem. This problem is to choose the carton and pallet dimensions so that the volume of product packed in a container is maximized. It requires a "one-shot" analysis to determine the solution.

The second is called the distributor's pallet packing problem. In this problem, the distributor fills an order from a customer. The problem is to pack the cartons on a standard pallet so as to maximize the volume placed on each pallet, thereby, minimizing the number of pallets used to ship the order. The problem addressed in this research is a constrained version of the distributor's pallet packing problem. This is because the number of cartons of each size that can be loaded on a pallet is restricted.

Due to difficulty of expanding existing two-dimensional loading methods described in Chapter II, two new three-dimensional pallet loading procedures have been developed in this research. One is a mixed 0-1 integer programming model which generates an exact optimal solution. This solution procedure may be time-consuming. The other is a heuristic dynamic programming algorithm which may give only "good" solution but requires less computation time when compared with the mixed 0-1 model.

The solution of the mixed 0-1 model explicitly gives the desired number of boxes of each size and the x, y, z coordinates of each box's

placement location on the pallet. An existing branch-and-bound technique is employed to solve the mixed 0-1 integer programming model. An example is also presented to illustrate the procedure of model transformation.

Unlike traditional dynamic programming, the heuristic dynamic programming algorithm involves the achievement of two goals. The first goal is to maximize the utilization of a pallet cube. The second goal is to make the box proportion satisfy some user-specified number. Post-solution adjustment may be also required for the heuristic dynamic programming procedure to obtain desired box proportions. These two algorithms are described in the following sections.

B. The Mixed 0-1 Integer Programming Model

1. Constraints of placement location on the pallet

Consider a set $B = \{b_1, b_2, \ldots, b_n\}$. B is a collection of n boxes. Each box b_i has length l_i, width w_i, and height h_i. A loading of B into a pallet of length L, width W and stacking height limit H is an assignment of boxes to a position within the pallet such that:

- no two boxes in the pallet overlap;
- each box is contained entirely within the pallet, with its sides parallel to the sides of the pallet;
- the proportion of the number of boxes of a given type to the total number of boxes of a full pallet load satisfies some user's specification.

An optimal loading is achieved if the use of pallet space is maximized under the consideration of the above three restraints. The stabili-

ty[1] (interlock between boxes) of a unit load built up on a pallet is not taken into consideration in this research.

Boxes in set B may or may not have the same dimensions. Also, the orientations of the boxes in set B are fixed permanently. The length, width and height of a box must be aligned with the length, width and height of the pallet, respectively. Length is defined as the dimension along the x-axis. Width is the dimension along the y-axis, and height is the dimension along the z-axis in Cartesian coordinate space. In this research, the orientation of box height is assumed to be fixed ("this side up"). Only the length and the width of a box are interchangeable. Thus, a box can be placed either in l x w x h or w x l x h direction on the pallet. Two boxes representing these two orientations must be individually included in set B. In addition, the placement location of a box in Cartesian coordinate space is measured relative to the front-bottom-left corner of the box throughout this chapter.

The following paragraphs describe the procedure of converting the requirements of avoiding box overlaps to mathematical constraints. Consider two partially overlapped boxes, A and B, as shown in Figure 3.1. The projections of these two boxes on the x-y, the x-z and the y-z planes are illustrated in Figures 3.2a, 3.2b and 3.2c, respectively.

---

[1] In industrial applications, the strapping and wrapping machines can be employed to secure a pallet load. See reference [66] for more information of various approaches for load security.

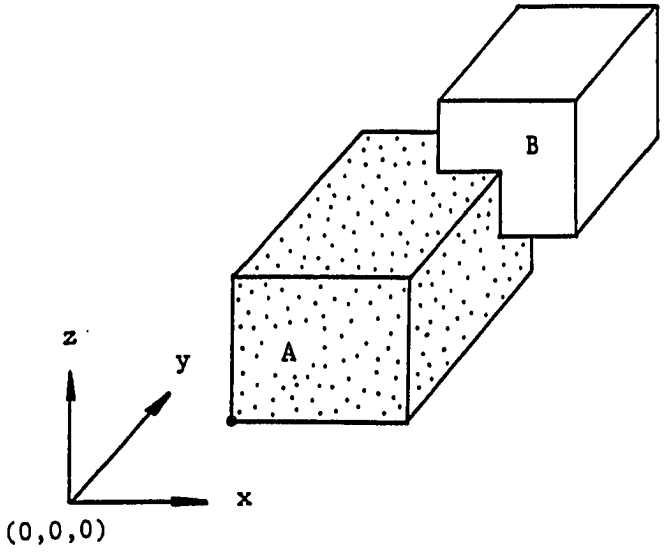


Figure 3.1. Two overlapped boxes

Suppose the location of box A is fixed, and that box B is free to move arbitrarily in Cartesian coordinate space. To avoid overlap of these two boxes, the following conditions must be satisfied.

Denote l_A, l_B = lengths of boxes A and B, respectively;

w_A, w_B = widths of boxes A and B;

h_A, h_B = heights of boxes A and B;

(x_A, y_A, z_A) = front-bottom-left corner coordinate of box A;

(x_B, y_B, z_B) = front-bottom-left corner coordinate of box B.

When considering the x-y plane (see Figure 3.2a):

$$x_B \geq x_A + l_A$$

or

$$x_B + l_B \geq x_A$$

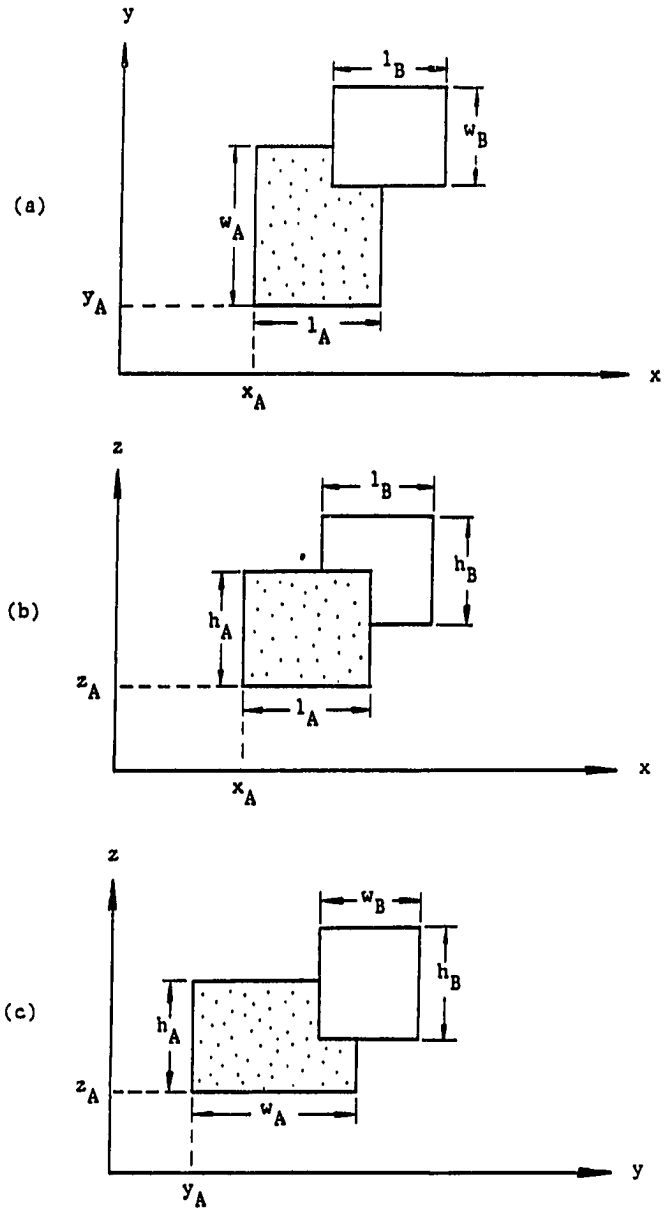


Figure 3.2. The projections of boxes on planes

or

$$y_B \geq y_A + w_A$$

or

$$y_B + w_B \leq y_A$$

When considering the x-z plane (see Figure 3.2b):

$$x_B \geq x_A + l_A$$

or

$$x_B + l_B \leq x_A$$

or

$$z_B \geq z_A + h_A$$

or

$$z_B + h_B \leq z_A$$

When considering the y-z plane (see Figure 3.2c):

$$y_B \geq y_A + w_A$$

or

$$y_B + w_B \leq y_A$$

or

$$z_B \geq z_A + h_A$$

or

$$z_B + h_B \leq z_A$$

By eliminating the redundant inequalities and rearranging the variables, equations (3.1) through (3.6) below are obtained:

$$x_B - x_A \geq l_A \tag{3.1}$$

or

$$x_A - x_B \geq l_B \tag{3.2}$$

or

$$y_B - y_A \geq w_A \tag{3.3}$$

or

$$y_A - y_B \geq w_B \tag{3.4}$$

or

$$z_B - z_A \geq h_A \tag{3.5}$$

or

$$z_A - z_B \geq h_B \tag{3.6}$$

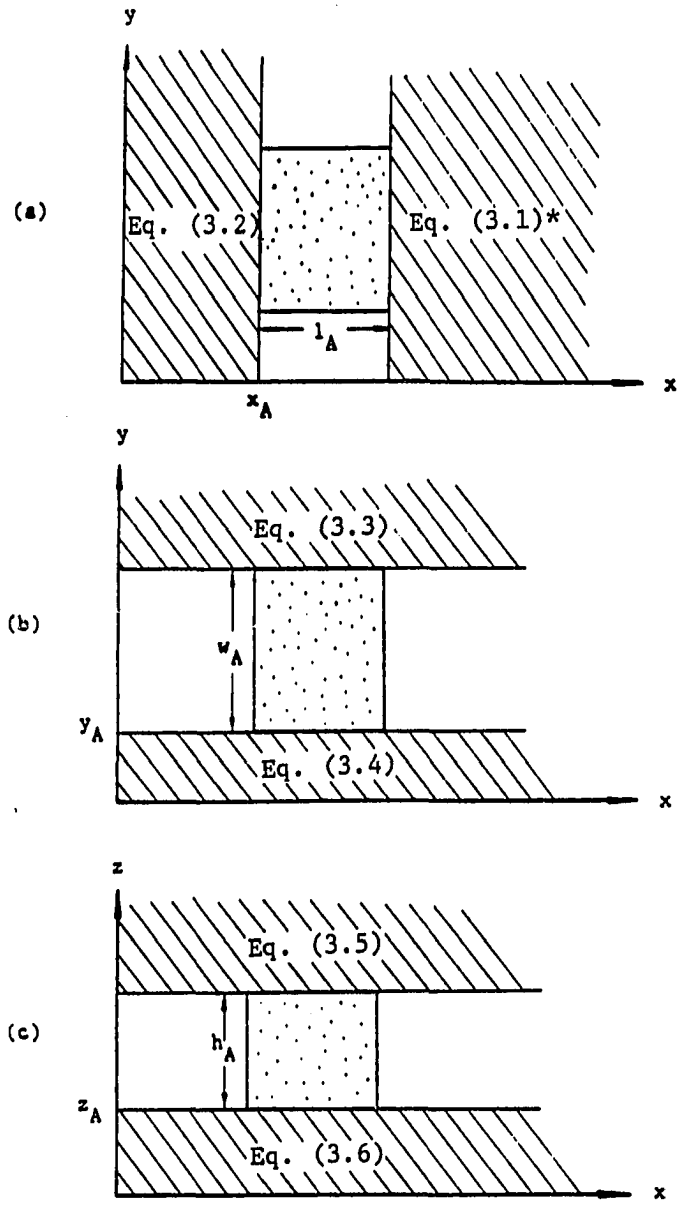
At least one of these six constraints must hold to prevent overlap of these two boxes. The safety regions in which box B can be placed are shown as the shaded areas in Figures 3.3a, 3.3b and 3.3c. These six regions correspond to the six constraints of eqs. (3.1)-(3.6).

For the set $B = \{b_1, b_2, \ldots, b_n\}$, all position relationships among these n boxes must be considered. The position constraints can be determined by establishing the constraints specified by eqs. (3.1)-(3.6). Since only two boxes are considered in each constraint set of eqs. (3.1)-(3.6), there are $\binom{n}{2} = n(n-1)/2$ combinations to be formulated.

A two-dimensional pallet packing problem is formulated first since the geometry of the problem can give an insight into the three-dimensional algorithm. The formulation of the three-dimensional packing problem is then expanded from the two-dimensional case.

2. Formulation of two-dimensional pallet packing

Consider a pallet of size L x W, and a collection of n rectangular pieces $B = \{b_1, b_2, \ldots, b_n\}$. Piece b_i in set B has length l_i and width w_i. This two-dimensional packing problem can be formulated as a mathematical programming model. Denote this as model I. The model may be



*Corresponding restraint equations.

Figure 3.3. The safety regions for Box B

expressed as:

Model I:

$$\text{Maximize} \quad Z = \sum_{k=1}^{n} a_k P_k \tag{3.7}$$

subject to

$$x_i - x_j \geq l_j \tag{3.8}$$

or

$$x_j - x_i \geq l_i \tag{3.9}$$

or

$$y_i - y_j \geq w_j \tag{3.10}$$

or

$$y_j - y_i \geq w_i \tag{3.11}$$

and

$$x_k \geq X^o P_k \tag{3.12}$$

$$y_k \geq Y^o P_k \tag{3.13}$$

$$x_k \leq (X^o+L)-l_k \tag{3.14}$$

$$y_k \leq (Y^o+W)-w_k \tag{3.15}$$

$$\sum_{m \in C_g} P_m \leq R_g \left(\sum_{m=1}^{n} P_m \right) \tag{3.16}$$

$$x_k, y_k \geq 0,$$

$$P_k \in \{0,1\},$$

$$i = 1,2,\ldots,n-1$$

$$j = i+1, i+2, \ldots, n$$

$$k = 1,2,\ldots,n,$$

$$g = 1,2,\ldots,r.$$

where: R_g = desired proportion of size g pieces;

a_k = the area of piece k = $l_k \cdot w_k$; piece k is the kth element in set B;

x_i = the x-coordinate of the bottom-left corner of piece i;

y_i = the y-coordinate of the bottom-left corner of piece i;

r = total number of box sizes.

(X^o,Y^o) in model I is the coordinate of the pallet's bottom-left corner. (X^o,Y^o) is selected in the x-y plane so that every piece in set B can be placed entirely within the large rectangle of length (X^o+L) and width (Y^o+W). The allocation of the pallet at coordinate (X^o,Y^o) also guarantees that the continuous variables, x_i and y_i, will result in positive solution. The restriction of the nonnegativity conditions on the decision variables are required for linear programming.

The values of X^o and Y^o can be determined using the following expressions.

$$X^o = n \cdot \max(l_i),$$

$$Y^o = n \cdot \max(w_i).$$

Figure 3.4 shows the location of the pallet in the x-y plane.

Eqs. (3.8)-(3.11) set up the position constraints such that no pieces will overlap with one another. These constraints are established according to eqs. (3.1)-(3.6). Since the packing problem in

question is two-dimensional, the height constraints (eqs. (3.5) and (3.6)) are not considered in model I.

Constraints (3.12)-(3.15) confine the placement boundary of a box on the pallet. P_k is a binary variable, which can only be either zero or one. It is associated with the kth piece in set B. When $P_k = 1$, the following must hold for eqs. (3.12)-(3.15).

$$x_k \geq X^o$$

$$y_k \geq Y^o$$

$$x_k \leq (X^o+L)-l_k$$

$$y_k \leq (Y^o+W)-w_k$$

This indicates that piece k is placed in the valid pallet area (shaded area ABCD in Figure 3.4). Thus, $P_k = 1$ represents the placement of piece k entirely within the pallet. Because the objective function is to maximize the total area covered on the pallet, this will force P_k to take on as many 1's as possible. Eqs. (3.12)-(3.15) become the following if all P_k's are set equal to 1.

$$x_k \geq X^o$$

$$y_k \geq Y^o$$

$$x_k \leq (X^o+L)-l_k$$

$$y_k \leq (Y^o+W)-w_k, \qquad k = 1,2,\ldots,n.$$

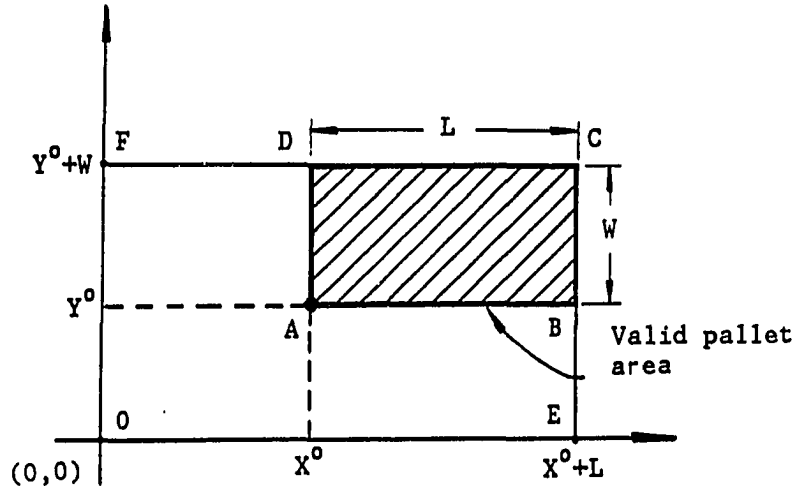


Figure 3.4. Pallet position in the x-y plane

The above relationships infer that all pieces in set B are placed within the small shaded area ABCD (see Figure 3.4), which represents the valid area of the pallet. However, due to the position constraints of eqs. (3.8)-(3.11), not all of the P_k's can be set equal to 1. That is, not all pieces can be placed onto the pallet (shaded area ABCD in Figure 3.4). Some P_k's will, therefore, be set equal to 0. This relaxes the restrictions of eqs. (3.12) and (3.13), and changes eqs. (3.12)-(3.15) as follows:

$$x_k \geq 0$$

$$y_k \geq 0$$

$$x_k \leq (X^o+L) - l_k$$

$$y_k \leq (Y^o+W) - w_k$$

This allows piece k to be placed in the large rectangle OECF (see Figure 3.4), but outside the valid shaded area ABCD.

Note that so long as one of the constraints (3.12) and (3.13) is not in effect, P_k will be set equal to zero. Therefore, $P_k = 0$ represents the placement of piece k outside the pallet either entirely or partially.

Eq. (3.16) defines the box proportion constraints. It states that the ratio of the number of size g pieces to the total number of pieces on the pallet must not exceed the user-specified proportion value R_g. In eq. (3.16), C_g is a subset of B. It consists of all pieces of size g regardless of the orientation, i.e.,

$$C_g = \{b_k \mid (l_k,w_k) = (l_g,w_g) \text{ or } (w_g,l_g), \quad 1 \leq k \leq n\}.$$

By solving this formulated model, the objective function maximizes the total area covered on the pallet by the pieces. In the final solution, any piece having a P_k value of zero will be discarded. The solution determines not only the required number of pieces of each type, but also the placement location of every piece on the pallet. The optimal pallet pattern is thus explicitly obtained from the final solution.

Eqs. (3.8)-(3.11) make the problem one of multiple choice programming [52]. By introducing additional binary variables, Hillier [55] has

transformed the multiple choice programming problem into a standard mixed 0-1 integer programming model. His transformation procedure is as follows:

Consider the case where, given a set of four possible constraints, it is only required that one of these constraints must hold. For eqs. (3.8)-(3.11), these restraints are:

$$x_j - x_i \leq -l_j$$

or

$$x_i - x_j \leq -l_i$$

or

$$y_j - y_i \leq -w_j$$

or

$$y_i - y_j \leq -w_i$$

An equivalent formulation of this requirement is

$$x_j - x_i \leq -l_j + Mu_1 \tag{3.17}$$

$$x_i - x_j \leq -l_i + Mu_2 \tag{3.18}$$

$$y_j - y_i \leq -w_j + Mu_3 \tag{3.19}$$

$$y_i - y_j \leq -w_i + Mu_4 \tag{3.20}$$

$$u_1 + u_2 + u_3 + u_4 = 3 \tag{3.21}$$

$$u_i \in \{0,1\},\ i=1,2,3,4.$$

where M is an extremely large number. The constraint on the u_j (eq.

3.21) guarantees that one of these artificial variables will equal zero and those remaining will equal 1. Therefore, one of the original constraints will be unchanged, and the rest will not be in effect because of the extremely large value of the right-hand-side. For instance, let $u_1 = 0$, $u_2 = u_3 = u_4 = 1$. Then, eq. (3.17) must be satisfied, and eqs. (3.18), (3.19) and (3.20) are relaxed due to the extremely large value of M.

Based on Hillier's procedure, each set of four multiple choice constraints requires four binary variables. By considering the following binary coding logic, the number of binary variables can be further reduced to two without any additional constraint. The four possible combinations of two binary variables u_1 and u_2 are

u_1	u_2
0	0
0	1
1	0
1	1

These four combinations can be used to set up the right-hand-side (RHS) values of the four multiple choice constraints. Eqs. (3.17)-(3.21) are equivalent to

$$x_j - x_i \leq -l_j + M(u_1+u_2) \tag{3.22}$$

$$x_i - x_j \leq -l_i + M[1 - (u_2-u_1)] \tag{3.23}$$

$$y_j - y_i \leq -w_j + M[1 - (u_1-u_2)] \tag{3.24}$$

$$y_i - y_j \leq -w_i + M[2 - (u_1+u_2)] \qquad (3.25)$$

$$u_1,\ u_2\ \varepsilon\ \{0,1\}$$

where M is an extremely large number.

The possible combinations of u_1 and u_2 and their corresponding constraint equations are presented in Table 3.1.

Table 3.1. Binary values and associated constraint equations

Binary vars.		RHS values of equations				Applicable constraint equation
u_1	u_2	(3.22)	(3.23)	(3.24)	(3.25)	
0	0	$-l_j$	M	M	2M	(3.22)
0	1	M	$-l_i$	2M	M	(3.23)
1	0	M	2M	$-w_j$	M	(3.24)
1	1	2M	M	M	$-w_i$	(3.25)

Since M is extremely large, the resulting right-hand-side value of $-l_i + M$ or $-w_i + M$ is simply indicated as M in Table 3.1. Table 3.1 shows that eqs. (3.22)-(3.25) are equivalent to the multiple choice constraints of eqs. (3.8)-(3.11). Therefore, model I can be reformulated as a standard mixed 0-1 integer programming model. Let this new formulation be model II as expressed below.

Model II:

$$\text{Maximize } Z = \sum_{k=1}^{n} a_k P_k$$

subject to

$$x_j - x_i \leq -l_j + M(u_{t1} + u_{t2})$$

$$x_i - x_j \leq -l_i + M[1 - (u_{t2} - u_{t1})]$$

$$y_j - y_i \leq -w_j + M[1 - (u_{t1} - u_{t2})]$$

$$y_i - y_j \leq -w_i + M[2 - (u_{t1} + u_{t2})]$$

$$x_k \geq X^o P_k$$

$$y_k \geq Y^o P_k$$

$$x_k \leq (X^o + L) - l_k$$

$$y_k \leq (Y^o + W) - w_k$$

$$\sum_{m \in C_g} P_m \leq R_g \cdot \left(\sum_{m=1}^{n} P_m \right)$$

$$x_k, y_k \geq 0$$

$$P_k, u_{t1}, u_{t2} \in \{0,1\}$$

$$i = 1,2,\ldots,n-1$$

$$j = i+1, i+2, \ldots, n$$

$$k = 1,2,\ldots,n$$

$$g = 1,2,\ldots,r$$

$$t = \begin{cases} (j-i), \text{ for } i=1 \\ (j-i) + \sum_{s=1}^{i-1} (n-s), \text{ for } i > 1 \end{cases}$$

where u_{t1} and u_{t2} are the artificial binary variables for converting the multiple choice constraints of set t to the standard "AND" constraints, $t = 1,2,\ldots,\binom{n}{2}$.

Note that the maximum value of any variable x_k is $(X^o+L) - l_k$ (see Eq. 3.14), and the minimum value of x_k is zero (see eq. 3.12 with $P_k = 0$). The maximum absolute difference between two variables x_i and x_j is $(X^o+L) - l_i$ or $(X^o+L) - l_j$. Likewise, the maximum absolute difference between variables y_i and y_j is $(Y^o+W) - w_i$ or $(Y^o+W) - w_j$. Therefore, the value of M will be sufficiently large by assigning

$$M \geq 2*\max\{X^o+L, Y^o+W\}.$$

Model II is a complete model for the two-dimensional packing problem. By introducing the height constraints of eqs. (3.5) and (3.6) to model II, the two-dimensional pallet packing formulation is extendable to a three-dimensional model. The following section describes the formulation procedure for the three-dimensional pallet packing problem.

3. Formulation of three-dimensional pallet packing problem

Let set $B = \{b_1, b_2, \ldots, b_n\}$ be a collection of n boxes. Each box b_i has length l_i, width w_i and height h_i. Boxes in set B are to be loaded onto a pallet of length L and width W with a stacking height

limit H.

a. Binary variables and multiple choice constraints Based on Hillier's formulation procedure [55], a set of the three-dimensional position constraints, eqs. (3.1)-(3.6), is equivalent to

$$x_j - x_i \leq -l_j + Mu_1 \tag{3.26}$$

$$x_i - x_j \leq -l_i + Mu_2 \tag{3.27}$$

$$y_j - y_i \leq -w_j + Mu_3 \tag{3.28}$$

$$y_i - y_j \leq -w_i + Mu_4 \tag{3.29}$$

$$z_j - z_i \leq -h_j + Mu_5 \tag{3.30}$$

$$z_i - z_j \leq -h_i + Mu_6 \tag{3.31}$$

and

$$u_1 + u_2 + u_3 + u_4 + u_5 + u_6 = 5 \tag{3.32}$$

$$u_i \in \{0,1\},\ i = 1,2,3,4,5,6.$$

where u_i's are binary variables, and M is an extremely large number.

Hillier's procedure requires six additional binary variables for each set of six multiple choice constraints. The binary variables can be further reduced to three with one additional constraint according to the following binary coding procedure. Let u_1, u_2 and u_3 be binary variables and

$$1 \leq u_1 + u_2 + u_3 \leq 2.$$

The six possible combinations of different binary values are

u_1	u_2	u_3
1	0	0
0	1	0
0	0	1
1	1	0
0	1	1
1	0	1

The above six combinations can be used to set up the right-hand-side values of six multiple choice constraints. The multiple choice constraints of eqs. (3.1)-(3.6) are equivalent to

$$x_j - x_i \leq -l_j + M(u_2 + u_3) \tag{3.33}$$

$$x_i - x_j \leq -l_i + M(u_1 + u_3) \tag{3.34}$$

$$y_j - y_i \leq -w_j + M(u_1 + u_2) \tag{3.35}$$

$$y_i - y_j \leq -w_i + M[2 - (u_1+u_2)] \tag{3.36}$$

$$z_j - z_i \leq -h_j + M[2 - (u_2+u_3)] \tag{3.37}$$

$$z_i - z_j \leq -h_i + M[2 - (u_1+u_3)] \tag{3.38}$$

and

$$u_1 + u_2 + u_3 \leq 2 \tag{3.39}$$

$$u_1 + u_2 + u_3 \geq 1 \tag{3.40}$$

$$u_1, u_2, u_3 \in \{0,1\}$$

Where M is an extremely large positive number.

Constraints (3.39) and (3.40) eliminate two possible combinations of $(U_1,U_2,U_3) = (0,0,0)$ and $(1,1,1)$. The binary values and their corresponding RHS values are presented in Table 3.2.

Table 3.2. Binary variables and associated RHS values

Binary vars.			RHS values of equations						Applicable constraint equation
U_1	U_2	U_3	(3.33)	(3.34)	(3.35)	(3.36)	(3.37)	(3.38)	
1	0	0	-1_j	M	M	M	2M	M	(3.33)
0	1	0	M	-1_j	M	M	M	2M	(3.34)
0	0	1	M	M	$-w_j$	2M	M	M	(3.35)
1	1	0	M	M	2M	$-w_i$	M	M	(3.36)
0	1	1	2M	M	M	M	$-h_j$	M	(3.37)
1	0	1	M	2M	M	M	M	$-h_i$	(3.38)

Therefore, three binary variables and two additional constraints can generate six combinations to convert a set of six multiple choice constraints to the standard "AND" constraints. The RHS values $-1_i + M$, $-w_j + M$, etc., in Table 3.2 are simply represented by M or 2M because of the extremely large value of M.

b. Three dimensional model notation Suppose the front-bottom-left corner of the pallet is located at the coordinate (X^o,Y^o,Z^o) in Cartesian coordinate space. Figure 3.5 pictorically shows the location of the pallet. (X^o,Y^o,Z^o) is selected such that all boxes in set B can be completely placed into the large cube OABCDE (see Figure 3.5).

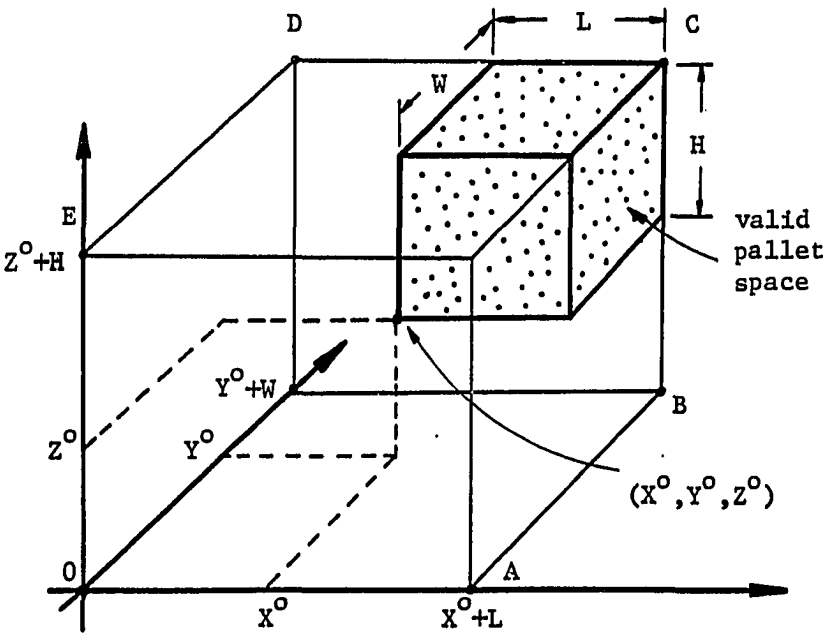


Figure 3.5. The pallet location in Cartesian coordinate space

Define the following symbols as:

B = a collection of n boxes to be considered
= $\{b_1,b_2,\ldots,b_n\}$

(l_i,w_i,h_i) = the dimensions of box i (b_i) in set B.
= length, width, and height, respectively.

(L,W,H) = the dimenions of a pallet
= length, width and height, respectively.

(X^O,Y^O,Z^O) = pallet location in Cartesian coordinate space along the x-, the y- and the z-axis, respectively.

The values of X^O,Y^O,Z^O may be determined using the expressions below.

$$X^O = n*\max_i(l_i),$$

$Y^o = n*\max_i(w_i)$,

$Z^o = n*\max_i(h_i)$.

(x_i,y_i,z_i) = decision variables
= the x, y, z coordinates of the placement location of the front-bottom-left corner of box i.

u_{t1},u_{t2},u_{t3} = artificial binary variables for converting six multiple choice constraints of set t to standard "AND" constraints, t = 1,2,..., $\binom{n}{2}$

P_k = a binary decision variable associated with the kth box in set B.
Load box k onto the pallet if $P_k = 1$.
Discard box k from set B if $P_k = 0$.

v_k = the volume of box k
= $l_i \cdot w_i \cdot h_i$

R_g = the desired box proportion of type g.

C_g = a subset of B; consists of all boxes of size g regardless of box orientation

= $\{b_k \mid (l_k,w_k,h_k) = (l_g,w_g,h_g)$ or

$(w_g,l_g,h_g),\ 1 \leq k \leq n\}$

M = an extremely large number, which can be selected to be twice of $\max\{X^o+L, Y^o+W, Z^o+H\}$ or larger.

r = total number of box types

The number of boxes of each type to be considered in set B can be determined using the following two equations.

$$n_g / \sum_{i=1}^{r} n_i = R_g \tag{3.41}$$

$$\sum_{i=1}^{r} n_i \cdot v_i = V \tag{3.42}$$

where n_g = the number of boxes of type g to be considered in set B

V = the volume of a pallet
= L·W·H

Eq. (3.41) states that the ratio of the number of type g boxes to the total number of boxes should equal R_g, the desired box proportion of type g. Eq. (3.42) indicates that the total cumulative box volumes should equal the pallet's volume. By solving eqs. (3.41) and (3.42), the number of boxes for type g can be obtained. The result is

$$n_g = \frac{R_g \cdot V}{\sum_{i=1}^{r} R_i v_i} \tag{3.43}$$

The solution obtained from (3.43) should be rounded up to the next higher integer value.

Each box with different orientations, either 1 x w x h or w x 1 x h, must be individually considered. For each type g, n_g boxes with orientation 1 x w x h and n_g boxes with orientation w x 1 x h may need to be included in set B.

c. Model formulation The three-dimensional pallet loading problem can now be formulated as a mixed 0-1 integer programming model. Let the three-dimensional formulation be Model III as shown below.

Model III:

$$\text{Maximize } Z = \sum_{k=1}^{n} v_k P_k \tag{3.44}$$

subject to

1) Avoid overlap of boxes

$$x_j - x_i \leq -l_j + M(u_{t2}+u_{t3}) \tag{3.45}$$

$$x_i - x_j \leq -l_i + M(u_{t1}+u_{t3}) \tag{3.46}$$

$$y_j - y_i \leq -w_j + M(u_{t1}+u_{t2}) \tag{3.47}$$

$$y_i - y_j \leq -w_i + M[2 - (u_{t1}+u_{t2})] \tag{3.48}$$

$$z_j - z_i \leq -h_j + M[2 - (u_{t2}+u_{t3})] \tag{3.49}$$

$$z_i - z_j \leq -h_i + M[2 - (u_{t1}+u_{t3})] \tag{3.50}$$

$$u_{t1} + u_{t2} + u_{t3} \leq 2 \tag{3.51}$$

$$u_{t1} + u_{t2} + u_{t3} \geq 1 \tag{3.52}$$

2) Confine placement boundary

$$x_k \geq X^o P_k \tag{3.53}$$

$$y_k \geq Y^o P_k \tag{3.54}$$

$$z_k \geq Z^o P_k \tag{3.55}$$

$$x_k \leq (X^o+L) - l_k \tag{3.56}$$

$$y_k \leq (Y^o+W) - w_k \tag{3.57}$$

$$z_k \leq (Z^o+H) - h_k \tag{3.58}$$

3) Satisfy desired box proportion

$$\sum_{m \in C_g} P_m \leq R_g \cdot \left(\sum_{m=1}^{n} P_m \right) \tag{3.59}$$

$$P_k, u_{t1}, u_{t2}, u_{t3} \in \{0,1\}$$

$$x_k, y_k, z_k \geq 0$$

$$i = 1,2,\ldots,n-1$$

$$j = i+1, i+2, \ldots, n$$

$$k = 1,2,\ldots,n$$

$$g = 1,2,\ldots,r$$

$$t = \begin{cases} (j-i), \text{ for } i=1 \\ (j-i) + \sum_{s=1}^{i-1} (n-s), \text{ for } i > 1 \end{cases}$$

The objective function, eq. (3.44), maximizes the total pallet volume occupied by boxes to be loaded. Based on George and Robinson's heuristic loading criteria [38], a box with larger volume is hard to fit late in the packing sequence. It is preferable to load the larger box early. The Simplex method of LP tends to make the variable having larger objective function coefficient enter first to the basis. This happens to be equivalent to the heuristic loading by first placing larger boxes onto the pallet, and then filling the remaining pallet

space with smaller boxes. The structure of this objective function may result in less computation time to reach an optimal solution.

Eqs. (3.45)-(3.52) set up the position constraints to ensure that no two boxes overlap. This is based on the result of eqs. (3.33)-(3.40). Eqs. (3.53)-(3.58) confine the placement boundary of boxes. The objective function, which maximizes the total volume occupied in the pallet cube, will force as many boxes as possible to be loaded onto the valid pallet space (see Figure 3.5). Whenever a box k is placed entirely into the valid pallet space, the associated binary variable, P_k, is set equal to one. Otherwise, P_k is set equal to zero. Finally, eq. (3.59) ensures that the total number of boxes of a given type will not exceed R_g proportion of the total number of boxes in a pallet.

In the final solution, any box having $P_k = 1$ is used to compose a pallet pattern. Any box having $P_k = 0$ is discarded from consideration. This formulated mixed 0-1 integer programming model for three-dimensional pallet loading problem thus gives the required number of boxes of each type, i.e., every box whose associated P_k equals one. It also generates the exact placement location of a box on the pallet, namely the coordinate (x_k, y_k, z_k).

4. A numerical example

Consider a distribution warehouse using 24" x 36" pallets for shipment. The stacking height limit of a pallet load is 10". A customer order requires 100 units of product A and 200 units of product B. Product A is packaged using the 24" x 24" carton, and product B

using the 12" x 10" carton. It is assumed that cartons A and B have identical heights of 8" so that both model II (two-dimensional problem) and model III (three-dimensional problem) can be illustrated using this example.

a. Two-dimensional model With model II, a collection of cartons, set B, and the location of the pallet in the x-y plane, (X^o,Y^o), must be determined.

Let $B = \{b_1,b_2,b_3\}$,

$(X^o,Y^o) = (100,100)$,

$M = 500$,

where b_1 = 24" x 24",

b_2 = 12" x 10",

b_3 = 10" x 12".

The carton proportions are obtained by calculating the order quantity.

For product A, $R_1 = 100/(100+200) = 1/3$.

For product B, $R_2 = 200/(100+200) = 2/3$.

The notation used for this example is listed below.

Box b_i	Product	Length l_i	Width w_i	Area a_i	Coordinate (x_i,y_i)	P_i
b_1	A	24	24	576	(x_1,y_1)	P_1
b_2	B	12	10	120	(x_2,y_2)	P_2
b_3	B	10	12	120	(X_3,y_3)	P_3

Therefore, this packing problem is equivalent to the following mixed 0-1 integer programming problem by using model II.

$$\text{Maximize } Z = 576P_1 + 120P_2 + 120P_3$$

subject to

(a)
$$x_2 - x_1 \leq -12 + 500(u_{11} + u_{12})$$
$$x_1 - x_2 \leq -24 + 500[1 - (u_{12}-u_{11})]$$
$$y_2 - y_1 \leq -10 + 500[1 - (u_{11}-u_{12})] \quad \text{(for } b_1 \text{ and } b_2\text{)}$$
$$y_1 - y_2 \leq -24 + 500[2 - (u_{11}+u_{12})]$$

(b)
$$x_3 - x_1 \leq -10 + 500(u_{21}+u_{22})$$
$$x_1 - x_3 \leq -24 + 500[1 - (u_{22}-u_{21})]$$
$$y_3 - y_1 \leq -12+500[1 - (u_{21}-u_{22})] \quad \text{(for } b_1 \text{ and } b_3\text{)}$$
$$y_1 - y_3 \leq -24 + 500[2 - (u_{11}+u_{12})]$$

(c)
$$x_3 - x_2 \leq -10 + 500(u_{31}+u_{32})$$
$$x_2 - x_3 \leq -12 + 500[1 - (u_{32}-u_{31})]$$
$$y_3 - y_2 \leq -12 + 500[1 - (u_{31}-u_{32})] \quad \text{(for } b_2 \text{ and } b_3\text{)}$$
$$y_2 - y_3 \leq -10 + 500[2 - (u_{31}+u_{32})$$

(d)
$$x_1 \geq 100P_1$$
$$y_1 \geq 100P_1$$
$$x_1 \leq (100+24) - 24$$
$$y_1 \leq (100+36) - 24$$

$$
\text{(e)}\begin{cases} x_2 \geq 100P_2 \\ y_2 \geq 100P_2 \\ x_2 \leq (100+24) - 12 \\ y_2 \leq (100+36) - 10 \end{cases}
$$

$$
\text{(f)}\begin{cases} x_3 \geq 100P_3 \\ y_3 \geq 100P_3 \\ x_3 \leq (100+24) - 10 \\ y_3 \leq (100+36) - 12 \end{cases}
$$

$$
\text{(g)}\begin{cases} P_1 \leq (1/3)(P_1+P_2+P_3) \\ P_2+P_3 \leq (2/3)(P_1+P_2+P_3) \end{cases}
$$

$$u_{t1}, u_{t2}, \in \{0,1\}, \quad t=1,2,3$$

$$P_1, P_2, P_3 \in \{0,1\}$$

$$x_i, y_i \geq 0, \quad i=1,2,3$$

b. Three dimensional model With model III, the collection of cartons, set B, and the location of the pallet in Cartesian coordinate space, (X^o, Y^o, Z^o) are selected as follows:

$$B = \{b_1, b_2, b_3\},$$

$$(X^o, Y^o, Z^o) = (100, 100, 100),$$

$$M = 500$$

where b_1 = 24" x 24" x 8",

b_2 = 12" x 10" x 8",

b_3 = 10" x 12" x 8".

The carton proportions are 1/3 and 2/3 for the products A and B, respectively.

The notation used for this example is presented below.

Box b_i	Product	Length l_i	Width w_i	Height h_i	Volume v_i	Coordinate (x_i,y_i,z_i)	P_i
b_1	A	24	24	8	4608	(x_1,y_1,z_1)	P_1
b_2	B	12	10	8	960	(x_2,y_2,z_2)	P_2
b_3	B	10	12	8	960	(x_3,y_3,z_3)	P_3

The loading problem is equivalent to the following mixed 0-1 integer programming model by using model III.

$$\text{Maximize } Z = 4608P_1 + 960P_2 + 960P_3$$

subject to

$$\text{(a)} \begin{cases} x_2 - x_1 \le -12 + 500(u_{12}+u_{13}) \\ x_1 - x_2 \le -24 + 500(u_{11}+u_{13}) \\ y_2 - y_1 \le -10 + 500(u_{11}+u_{12}) \quad \text{for } b_1 \text{ and } b_2 \\ y_1 - y_2 \le -24 + 500[2 - (u_{11}+u_{12})] \\ z_2 - z_1 \le -8 + 500[2 - (u_{12}+u_{13})] \\ z_1 - z_2 \le -8 + 500[2 - (u_{11}+u_{13})] \\ 1 \le u_{11} + u_{12} + u_{13} \le 2 \end{cases}$$

(b)
$$
\begin{cases}
x_3 - x_1 \leq -10 + 500(u_{22}+u_{23}) \\
x_1 - x_3 \leq -24 + 500(u_{21}+u_{23}) \\
y_3 - y_1 \leq -12 + 500(u_{21}+u_{22}) \quad \text{for } b_1 \text{ and } b_3 \\
y_1 - y_3 \leq -24 + 500[2 - (u_{21}+u_{22})] \\
z_3 - z_1 \leq -8 + 500[2 - (u_{22}+u_{23})] \\
z_1 - z_3 \leq -8 + 500[2 - (u_{21}+u_{23})] \\
1 \leq u_{21} + u_{22} + u_{23} \leq 2
\end{cases}
$$

(c)
$$
\begin{cases}
x_3 - x_2 \leq -10 + 500(u_{32}+u_{33}) \\
x_2 - x_3 \leq -12 + 500(u_{31}+u_{33}) \\
y_3 - y_2 \leq -12 + 500(u_{31}+u_{32}) \quad \text{for } b_2 \text{ and } b_3 \\
y_2 - y_3 \leq -10 + 500[2 - (u_{31}+u_{32})] \\
z_3 - z_2 \leq -8 + 500[2 - (u_{32}+u_{33})] \\
z_2 - z_3 \leq -8 + 500[2 - (u_{31}+u_{33})] \\
1 \leq u_{31} + u_{32} + u_{33} \leq 2
\end{cases}
$$

(d)
$$\begin{cases} x_1 \geq 100P_1 \\ y_1 \geq 100P_1 \\ z_1 \geq 100P_1 \\ x_1 \leq (100+24) - 24 \\ y_1 \leq (100+36) - 24 \\ z_1 \leq (100+10) - 8 \end{cases}$$

(e)
$$\begin{cases} x_2 \geq 100P_2 \\ y_2 \geq 100P_2 \\ z_2 \geq 100P_2 \\ x_2 \leq (100+24) - 12 \\ y_2 \leq (100+36) - 10 \\ z_2 \leq (100+10) - 8 \end{cases}$$

(f)
$$\begin{cases} x_3 \geq 100P_3 \\ y_3 \geq 100P_3 \\ z_3 \geq 100P_3 \\ x_3 \leq (100+24) - 10 \\ y_3 \leq (100+36) - 12 \\ z_3 \leq (100+10) - 8 \end{cases}$$

(g)
$$\begin{cases} P_1 \leq (1/3)(P_1+P_2+P_3) \\ P_2 + P_3 \leq (2/3)(P_1+P_2+P_3) \end{cases}$$

$u_{t1}, u_{t2} u_{t3} \in \{0,1\}$, $t = 1,2,3$

$P_1, P_2, P_3 \in \{0,1\}$

$x_i, y_i, z_i \geq 0$, $i = 1,2,3$

For both two- and three-dimensional formulation models of this example, constraints in (a) set up the position restrictions for boxes 1 and 2 so that these two boxes will not overlap. Similarly, constraints in (b) and (c) are for boxes 1 and 3, and boxes 2 and 3, respectively. Under the restraints of (a), (b), and (c), it is guaranteed that no boxes will overlap.

Constraints in (d) indicate that box 1 (b_1) can only be placed in the large rectangle of size 124" x 136". When the resultant value of P_1 equals one, then box 1 is placed entirely in the valid pallet space starting from coordinate (100,100) and ending at coordinate (124,136) (see Figure 3.3). Pallet placement coordinate of (100,100) is selected to make sure that all boxes, b_1, b_2 and b_3 in set B can be loaded into the larger cube starting from (0,0) and ending at coordinate (124,136). This also guarantees that all decision variables will satisfy the requirement of nonnegative values. Similarly, constraints in (e) and (f) confine the placement boundary for box 2 (b_2) and box 3 (b_3), respectively.

Constraints in (g) represent that box proportions of products A and B must not exceed 1/3 and 2/3 of the total boxes loaded on the pallet, respectively.

One of the possible optimal solutions[1] using the two-dimensional formulated model is

$$(P_1,P_2,P_3) = (1,1,1)$$

$$(x_1,y_1) = (100,100)$$

$$(x_2,y_2) = (100,126)$$

$$(x_3,y_3) = (114,124)$$

$$(u_{11},u_{12}) = (1,1)$$

$$(u_{21},u_{22}) = (1,1)$$

$$(u_{31},u_{32}) = (0,1)$$

This solution gives the optimal objective function value of 816. Recall that initially the pallet is placed at coordinate $(X^o,Y^o) = (100,100)$. By subtracting (X^o,Y^o), the resulting placement location of a box, (x_i,y_i), can be converted back to the original coordinate. This gives

$$(x_1,y_1) = (0,0)$$

$$(x_2,y_2) = (0,26)$$

$$(x_3,y_3) = (14,24)$$

This solution corresponds to the optimal pallet pattern as shown in Figure 3.6.

---

[1]The solution procedure will be described in the next section. It takes 36 seconds to determine this optimal solution using an IBM AT microcomputer.

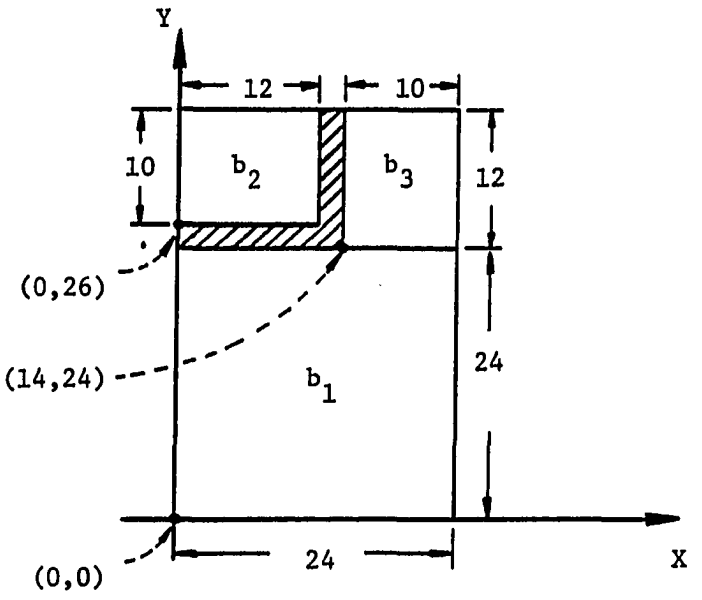


Figure 3.6. The optimal pallet pattern

5. A branch-and-bound solution procedure

The branch-and-bound principle has been established as a practical computational procedure for mixed integer programming since the early work of Lane and Doig [65]. The efficiency of a branch-and-bound algorithm comes from the ability to eliminate from consideration large subsets of potential optimal solutions. Two principal rules of eliminating such subsets are

(i) to show that no feasible completion of a given partial assignment exists; and

(ii) to show that no optimal completion of a given partial assignment exists.

a. Overview The branch-and-bound algorithm consists of a systematic search of continuous solutions in which the integer variables are successively forced to take on integer values. The algorithm starts by finding an optimal solution to the continuous problem (using LP procedure) where the integrality requirements are relaxed. If this solution is integer, then an optimal solution is reached. If not, then two subproblems are formed. Assume the binary variable u_k has continuous value at current solution stage. One subproblem thus has the constraint $u_k = 0$; and another subproblem has the constraint $u_k = 1$. The process of forming the subproblems is called branching. Each of these subproblems is solved again as a continuous problem. Each subproblem is considered as a node. All these nodes whose values of the objective function are lower than the current best solution found for a maximization problem are eliminated from a list of nodes. It is said that the corresponding nodes are fathomed. The search for the optimal solution terminates when all the nodes are fathomed. The current best feasible solution gives the optimal solution of the problem.

b. Algorithm efficiency strategies The efficiency of the branch-and-bound algorithm is based on the selection of the variable and the node to branch. Many strategies of choosing branching variables and nodes have been developed and surveyed in detail by Mitra [72] and Gupta and Ravindran [48]. Based on experiments of Gupta and Ravindran, "Most Fractional Integer Variable" and "Branch from the Node with Highest Bound" have performed acceptably for branch-and-bound procedure to se-

lect branching variables and nodes, respectively. Furthermore, these two strategies are simple and lend themselves to computer coding. They are applied in the research to a branch-and-bound algorithm for solving the formulated mixed 0-1 model. These two branching strategies are described in detail as follows:

• Most Fractional Integer Variable: This is used to select branching variables. This strategy selects the variable which is farthest from its nearest integer value. That is, choose a variable u_k from the set

$$\{u_k | k \in B1 \cap B2, \quad 0 < u_k < 1\}$$

which maximizes $\min\{u_k, 1-u_k\}$,

where B1 is a set of binary variables,

B2 is a set of indices of the basic variables in the continuous optimal solution at the current solution stage.

• Branch from the Node with Highest Bound: This is used to select branching node. In this strategy, the node which currently has the highest bound on the objective is selected for branching.

c. The Kochenberger's procedure Kochenberger and Richard [63] have developed a branch-and-bound procedure which employs linear programming techniques to solve for continuous solutions. The procedure for branch-and-bound will not destroy the primal feasibility of the LP solution. A brief description of the procedure is as follows:

The problem is to

$$\text{Maximize} \quad Z = \sum_{j=1}^{p} C_j u_j \tag{3.60}$$

subject to

$$\sum_{j=1}^{p} a_{ij}u_j + \sum_{j=p+1}^{p+n} a_{ij}x_j \leq b_i, \quad i=1,2,\ldots,m \tag{3.61}$$

$$u_j \in \{0,1\}, \quad j=1,2,\ldots,p \tag{3.62}$$

$$x_j \geq 0, \quad j=p+1,\ldots,p+n.$$

The coefficients of the objective function for each binary variable u_j may be rewritten as

$$C_j^* = C_j + M\delta_j$$

where M is an extremely large positive number. To make sure M is sufficiently large, the value of M can be selected to be twice of $n*\max_j\{C_j\}$ or larger. δ_j is a parameter that can be -1, 0, or +1. The new objective function is

$$\text{Maximize} \quad Z = \sum_{j=1}^{p} C_j^* u_j$$

Initially, all the δ_j are set equal to zero. By setting δ_j to -1 or +1, the u_j will be forced to be zero or one, respectively, because of the extremely large penalty (-M) or gains (+M). If such a result is not possible, it is concluded that no feasible solution exists with u_j at zero (or one) and fathom (eliminate from the list of subproblems) accordingly.

The Kochenberger's branch-and-bound procedure is presented, step by step, below.

STEP 1: Initialize. Set $\delta_j = 0$ for all j. Form the LP model of eqs. (3.60) and (3.61) with the relaxed constraint of eq. (3.62), i.e.,

$$u_j \leq 1, \quad j=1,2,\ldots,p.$$

Put this problem on the list. Denote Z* to be the best available lower bound on Z (the objective function value).

STEP 2: Select branching node. Choose the branching node from the list using the strategy of "Branch from the Node with Highest Bound". Terminate the procedure and deliver the optimal solution if the list is empty.

STEP 3: Solve the subproblem using LP. Starting with the current basis, perform normal Simplex method until the problem at hand is solved. Eliminate the node from the list and return to STEP 2, if

(i) the new objective function value is less than or equal to Z*; or

(ii) $0 < u_j < 1$ and its associated $\delta_j \neq 0$ (i.e., this solution is fathomed due to infeasibility.

Otherwise, go to STEP 4.

STEP 4: Check all binary restrictions. If the continuous LP solution generated in STEP 3 is binary, then record it and update Z*. Return to STEP 2. Otherise, go to STEP 5.

STEP 5: Partition. Select a fractional valued variable u_j using the strategy of "Most Fractional Integer Variable". Split the current LP model to create two new LP subproblems. In one subproblem, δ_j is set

to -1. In the other subproblem, δ_j is set to +1. Add these two subproblems to the list. Return to STEP 2.

Since δ_j is originally set to zero, the change of the δ_j value thus changes the coefficient of the objective function, i.e., the coefficient of binary variable u_j is changed from C_j to $C_j + M$ or $C_j - M$. The LP's at each node (subproblem) differ only in their cost coefficients. The technique of sensitivity analysis used in LP can be employed to solve the additional subproblems. Starting with the current LP basis, a given node problem can be solved by doing a few additional primal points without iterating the entire LP procedure all over again. No basis information need be stored for each node except the current basis. This effort minimizes computer memory requirements and computation time.

A BASIC program implementing Kochenberger's branch-and-bound solution procedure is presented in Appendix A. The example in the previous section has been solved using a BASIC program on an IBM AT microcomputer.

In addition to Kochenberger's branch-and-bound approach, there are many research papers which directly contribute to the formulated mixed 0-1 integer programming model. A survey of the literature related to the mixed 0-1 integer programming is given below.

d. Other computational approaches Ruthedge [81] has presented a Simplex method procedure to solve mixed 0-1 integer programming problem. Ohtake and Nishida [73], and Davis, Kendrick and Weitzman [27]

have employed branch-and-bound algorithms for mixed 0-1 integer programming. The reduced cost branch-and-bound algorithms using the Extended Control Language of the IBM MPSX/370-MIP/370 mixed integer programming package [70] have been developed by Martin et al. [69] and Martin and Schrage [68]. Côté and Laughton [25] has used partitioning method which separates the integer from the continuous variables. He then used heuristics to approach the large-scale mixed integer programming problems. Cooper and Farhangian [24], Healy [52] and Jeroslow and Lowe [61] have presented various approaches for solving the linear programming with multiple choice constraints.

e. Summary The formulated mixed 0-1 model does provide exact solutions for the pallet packing problem. However, use of 0-1 integer variables and multiple choice constraints may require extremely long computation times to reach final optimal solutions. A problem as simple as the previous example consists of nine (9) continuous variables, twelve (12) binary variables, and forty-four (44) constraints for the three-dimensional model. It is time-consuming to obtain the optimal pallet pattern, even with the use of a computer.

A new heuristic algorithm based on dynamic programming has been developed to overcome the shortcomings of the mixed 0-1 integer model. For the heuristic algorithm, there is trade-off between the quality of final solution and computation time. Only a "good" solution may be obtained. However, the computation time can be significantly reduced when compared with that of the formulated mixed 0-1 model. The heuristic

dynamic programming appraoch for the three-dimensional pallet packing problem is the subject of the following section.

C. The Heuristic Dynamic Programming Approach

The heuristic approach has been inspired from the early developmental work of Haims and Freeman [51]. Their algorithm is only applicable for two-dimensional pallet packing problems in which no restriction is placed on the number of small pieces of each size.

The new developed heuristic procedure is carried out with a dynamic programming (DP) algorithm to solve three-dimensional pallet packing problems. The algorithm involves a sequence of iterations, each of which allows a higher degree of packing. Each stage in the dynamic programming algorithm allocates one type of box.

Unlike traditional dynamic programming, there are two goals to be achieved for the heuristic DP algorithm. First, it is desired to place as many boxes as possible onto the pallet so as to maximize pallet space utilization. Secondly, for each box type, the number of boxes packed on a pallet (or box proportion) should not exceed some user-specified number. These two goals usually contradict each other. One has to compromise between these two goals to determine a "rational" solution. The recursive dynamic programming procedure and the criteria to determine a solution from these two conflicting goals are described in the sections that follow.

1. Maximization of pallet space usage (goal 1)

To apply the dynamic programming procedure, the dimensions of the pallet and boxes of all types are restricted to integer values. It is assumed that the orientations of boxes and pallet are predetermined and remain fixed. The lengths, widths and heights of boxes and pallet are parallel to each other, respectively. No interchange of orientations is allowed. Boxes of different orientations are considered to be a unique type even though they may have exactly the same dimensions. The procedure of goal 1 only tends to maximize utilization of a pallet cube without restriction on the number of boxes of each type.

Each box type is considered to be a stage in the dynamic programming algorithm. Also, the allocated box is placed in an index pallet, represented by (L_x,W_x,H_x). The dimensions of the index pallet are then increased by 1 at a time from a unit cube (1,1,1) to the pallet size (L,W,H).

Denote (L,W,H) = dimensions of a pallet;
= length, width and height, respectively

(l_i,w_i,h_i) = dimensions of a box of type i;
= length, width and height, respectively

v_i = volume of a box of type i
= $l_i \cdot w_i \cdot h_i$;

r = total number of box types.

Define the following allocation vectors:

$$\underline{a}_1(L_x,W_x,H_x) = (a_1)$$

$$\underline{a}_2(L_x,W_x,H_x) = (a_1,a_2)$$

$$\vdots$$

$$\underline{a}_r(L_x,W_x,H_x) = (a_1,a_2,\ldots,a_r)$$

$$\underline{a}_k(L_x,W_x,H_x) = \underline{0}, \text{ if any of the three arguments equals zero, } k=1,2,\ldots,r.$$

$\underline{a}_i(L_x,W_x,H_x)$ is an i-dimensional vector at stage i. The jth element of $\underline{a}_i$, a_j, represents the number of type j boxes allocated in the index pallet (L_x,W_x,H_x), where $i = 1,2,\ldots,r$ and $j = 1,2,\ldots,i$.

Also, define the following return function:

$$F_1(L_x,W_x,H_x) = \max(v_1a_1)$$

$$F_2(L_x,W_x,H_x) = \max(v_1a_1+v_2a_2)$$

$$\vdots$$

$$F_r(L_x,W_x,H_x) = \max(v_1a_1+\ldots+v_ra_r)$$

$$F_k(L_x,W_x,H_x) = 0, \text{ if any of the three arguments equals zero, } k=1,2,\ldots,r.$$

$F_i(L_x,W_x,H_x)$ is the maximum return function at stage i given the index pallet (L_x,W_x,H_x).

The index pallet (L_x,W_x,H_x) varies from (1,1,1) to (L,W,H), and increases by 1 at a time. Therefore, $F_r(L,W,H)$ is the final optimal function value to deliver. The term $\underline{a}_r(L,W,H)$ gives the required number of boxes of each type.

a. Haims and Freeman's two-dimensional allocation procedure

Consider first a two-dimensional case of allocation since the geometry of the problem can give an insight into the three-dimensional algorithm and so aids the understanding. Height (symbols H, H_x, and h_i) is ignored for the time being. The following two-dimensional dynamic programming procedure is adopted from the development of Haims and Freeman [51].

For all values of L_x, W_x such that $L_x > l_i$ and $W_x > w_i$, a box i can be allocated as shown in Figure 3.7.

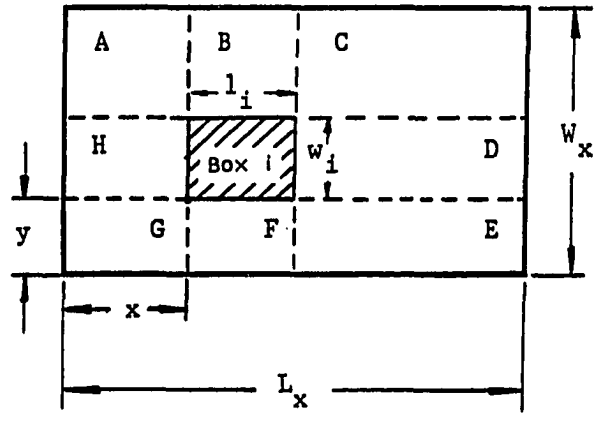


Figure 3.7. Allocation of a box i at (x,y)

Eight subrectangles, A, B, C, D, E, F, G and H, as defined in Figure 3.7 are created by allocating a box of type i at location (x,y). This generates sixteen (16) combinations of the subrectangles. One of the combinations is

$$A+B+C = L_x \cdot (W_x - w_i - y)$$

$$D+E = (L_x - l_i - x) \cdot (w_i + y)$$

$$F+G = (l_i + x) \cdot y$$

$$H = x \cdot w_i$$

This means four smaller index pallets A+B+C, D+E, F+G and H are generated from the original index pallet of size L_x by W_x when a box i is allocated at coordinate (x,y). The subindex pallet A+B+C has the dimensions of (L_x) by ($W_x - w_i - y$), D+E is ($L_x - l_i - x$) by ($w_i + y$), and so on. The combination of these four subindex pallets is shown in Figure 3.8.

Consider the subindex pallet A+B+C and two smaller index pallets A+B and C. The sum of areas covered on both A+B and C by pieces is at most as good as the subindex pallet A+B+C. A+B and C cannot generate any solution better than A+B+C. The subindex pallet A+B+C contains all

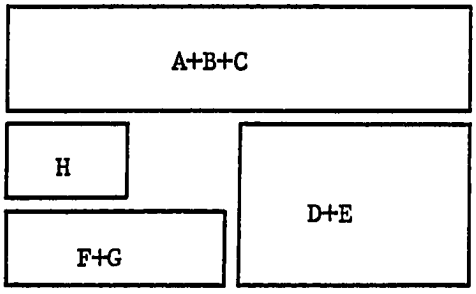


Figure 3.8. A combination of four subindex pallets

possible solutions of the two smaller index pallets A+B and C. Therefore, combinations like A+B and C is discarded from further consideration. Based on this logic, the sixteen possible combinations of subindex pallets are:

1) A+B+C
D+E
F+G
H

2) A+B+C
D+E
F
H+G

3) A+B+C
D
E+F
G+H

4) A+B+C
D
E+F+G
H

5) A+B
C+D
E+F
H+G

6) A+B
C+D+E
F+G
H

7) A+B
C+D+E
F
G+H

8) A+B
C+D
E+F+G
H

9) A+H
G+F
E+D
B+C

10) A+H
G+F+E
D+E
B

11) A+H
G+F+E
D
B+C

12) A+H
G+F
C+D+E
B

13) A+H+G
B+C
D
E+F

14) A+H+G
B+C
D+E
F

15) A+H+G
B
C+D
E+F

16) A+H+G
B
C+D+E
F

Each subindex pallet defined in terms of the dimensions is listed below.

(See Figure 3.7.)

$$B = (l_i)\cdot(W_x - w_i - y)$$

$$D = (L_x - l_i - x)\cdot(w_i)$$

$$F = (l_i)\cdot(y)$$

$$H = (x)\cdot(w_i)$$

$$A+B = (l_i + x)\cdot(W_x - w_i - y)$$

$$A+H = (x)\cdot(W_x - y)$$

$$B+C = (L_x - x)\cdot(W_x - w_i - y)$$

$$C+D = (L_x - l_i - x)\cdot(W_x - y)$$

$$D+E = (L_x - l_i - x)\cdot(w_i + y)$$

$$E+F = (L_x - x)\cdot(y)$$

$$F+G = (l_i + x)\cdot(y)$$

$$H+G = (x)\cdot(w_i + y)$$

$$A+B+C = (L_x)\cdot(W_x - w_i - y)$$

$$A+H+G = (x)\cdot(W_x)$$

$$C+D+E = (L_x - l_i - x)\cdot(W_x)$$

$$E+F+G = (L_x)\cdot(y)$$

For every index pallet (L_x, W_x) currently considered, the maximum return function values of the above subindex pallets have been determined in the previous procedure since all these subindex pallets have smaller dimensions than those of the index pallet.

Since the pallet is symmetrical, it is only necessary to evaluate the discrete placement of a box of type i in one-fourth of the rectangle (L_x,W_x). Observe the allocations of box i in Figures 3.9a and 3.9b.

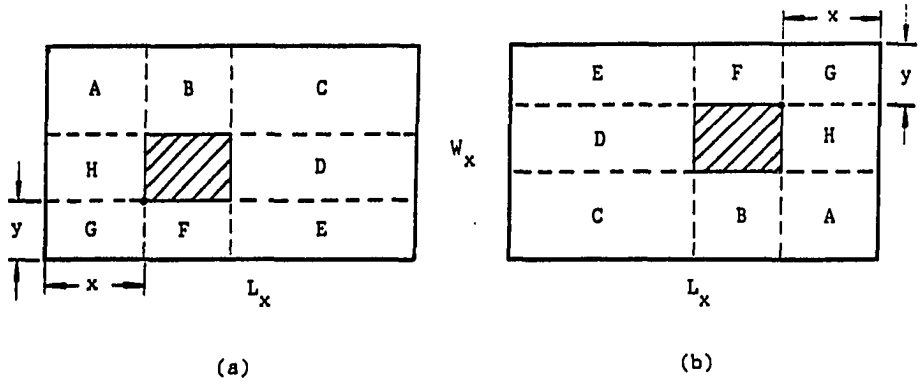


Figure 3.9. Symmetrical allocations

Because Figure 3.9b is equivalent to Figure 3.9a by rotating 180°, both patterns result in the same subrectangles. Therefore, it is only necessary to evaluate (x,y), the allocation of a box i in (L_x,W_x), from (1,1) to $([L_x/2],[W_x/2])$ increasing by 1 at a time.

For the two-dimensional case, there are sixteen combinations to be considered for each given value of (x,y). The objective is to select the combination with the maximum $F_i(L_x,W_x)$. The recursive function has the form

$$F_i(L_x,W_x) = \text{Max}\begin{cases} \text{1) Combination 1:} \\ \quad l_i \cdot w_i, \text{ the area of a box of type i} \\ \quad + F_i(L_x, W_x - w_i - y), \text{ the subindex pallet A+B+C} \\ \quad + F_i(L_x - l_i - x, w_i + y), \text{ the subindex pallet D+E} \\ \quad + F_i(l_i + x, y), \text{ the subindex pallet F+G} \\ \quad + F_i(x, w_i), \text{ the subindex pallet H} \\ \text{2) Combination 2:} \\ \quad l_i \cdot w_i + F_i(L_x, W_x - w_i - y) + F_i(L_x - l_i - x, w_i + y) \\ \quad + F_i(l_i, y) + F_i(x, w_i + y) \\ \quad \vdots \\ \text{16) Combination 16:} \\ \quad l_i \cdot w_i + F_i(x, W_x) + F_i(l_i, W_x - w_i - y) \\ \quad + F_i(L_x - l_i - x, W_x) + F_i(l_i, y) \\ \text{17) } F_{i-1}(L_x, W_x) \end{cases}$$

The maximum return function value of $F_i(L_x,W_x)$ is selected from one of the sixteen combinations of subindex pallets. In the instance that all return function values of these sixteen combinations are less than the maximum function value at the previous stage, the best previous function value is used for the current stage.

b. The three-dimensional allocation of boxes Now consider the three-dimensional case. Twenty-six sub-cubes are created when a box i is allocated at coordinate (x,y,z) of a three-dimensional index pallet of size (L_x,W_x,H_x). There may be hundreds of combinations of sub-index

pallets with these twenty-six sub-cubes. To reduce computation time, only six combinations of sub-index pallets are taken into consideration. Since not every possible combination of sub-index pallets is considered, only "sub-optimal" solutions may be obtained using this heuristic dynamic programming procedure. The allocation of a box i in the index pallet (L_x,W_x,H_x) at location (x,y,z) is shown in Figure 3.10. Also, the six combinations of sub-index pallets are illustrated in Figures 3.11a, b, c, d, e, and f.

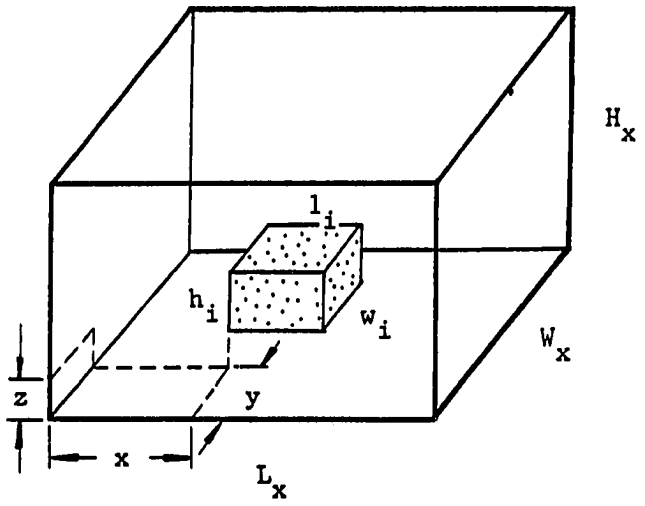


Figure 3.10. Allocation of a box in the index pallet

With an index pallet (L_x,W_x,H_x),

where $L_x = 1,2,\ldots,L$;

$W_x = 1,2,\ldots,W;$

$H_x = 1,2,\ldots,H,$

allocate a box i at coordinate (x,y,z) in terms of the front-bottom-left corner of the box. For every given index pallet (L_x,W_x,H_x) such that $L_x > l_i$, $W_x > w_i$ and $H_x > h_i$, the allocated coordinate (x,y,z) should be changed as follows:

$x = 1,2,\ldots, \min(L_x-l_i,[L_x/2]);$

$y = 1,2,\ldots, \min(W_x-w_i,[W_x/2]);$

$z = 1,2,\ldots, \min(H_x-h_i,[H_x/2).$

Then the return function and allocation vector are determined as shown below.

If $L_x < l_i$ or $W_x < w_i$ or $H_x < h_i$, then

$$F_i(L_x,W_x,H_x) = F_{i-1}(L_x,W_x,H_x), \text{ and}$$

$$\underline{a}_i(L_x,W_x,H_x) = \underline{a}_{i-1}(L_x,W_x,H_x).$$

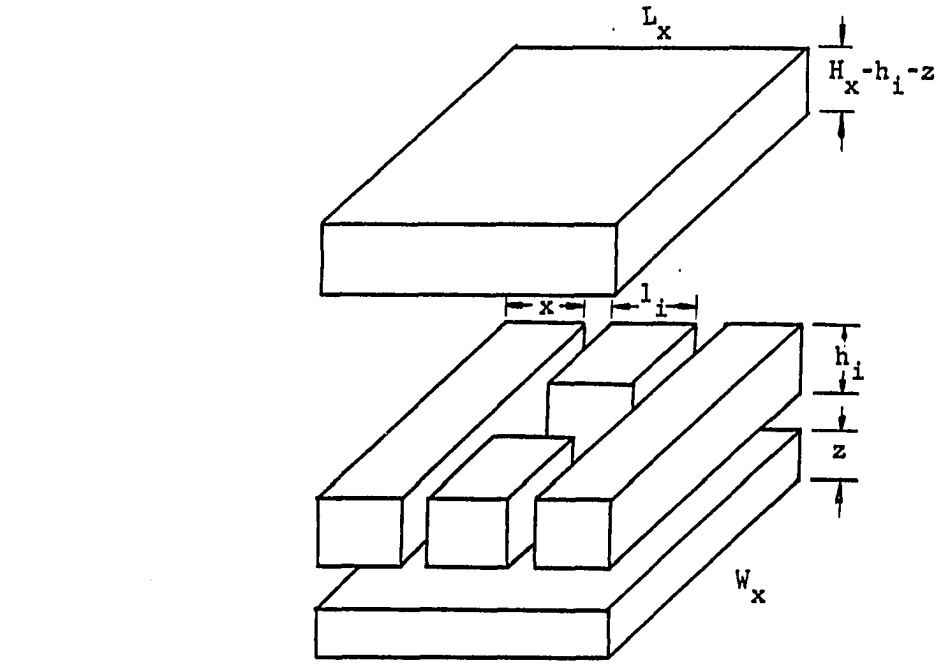
Otherwise,

$$F_i(L_x,W_x,H_x) = \max \begin{cases} 1)\ v_i + F_i(L_x,W_x,z) + F_i(L_x,W_x,H_x-h_i-z) \\ \quad + F_i(x,W_x,h_i) + F_i(L_x-l_i-x,W_x,h_i) \\ \quad + F_i(l_i,y,h_i) + F_i(l_i,W_x-w_i-y,h_i) \\ \quad \text{(see Fig. 3.11a)} \\ 2)\ v_i + F_i(L_x,W_x,z) + F_i(L_x,W_x,H_x-h_i-z) \\ \quad + F_i(x,w_i,h_i) + F_i(L_x-l_i-x,w_i,h_i) \\ \quad + F_i(L_x,y,h_i) + F_i(L_x,W_x-w_i-y,h_i) \\ \quad \text{(see Fig. 3.11b)} \end{cases} \tag{3.63}$$

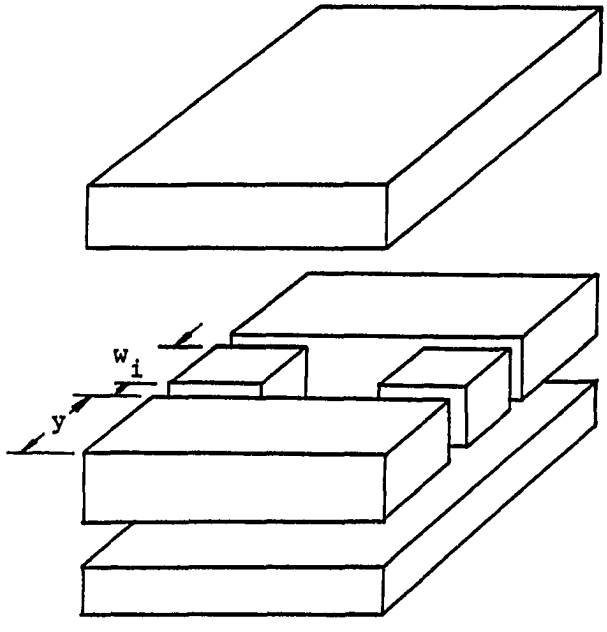
$$
\begin{cases}
3)\ v_i + F_i(x,W_x,H_x) + F_i(L_x-l_i-x,W_x,H_x) \\
\qquad + F_i(l_i,y,H_x) + F_i(l_i,W_x-w_i-y,H_x) \\
\qquad + F_i(l_i,w_i,H_x-h_i-z) + F_i(l_i,w_i,z) \\
\quad \text{(see Fig. 3.11c)} \\
4)\ v_i + F_i(x,W_x,H_x) + F_i(L_x-l_i-x,W_x,H_x) \\
\qquad + F_i(l_i,y,h_i) + F_i(l_i,W_x-w_i-y,h_i) \\
\qquad + F_i(l_i,W_x,H_x-h_i-z) + F_i(l_i,W_x,z) \\
\quad \text{(see Fig. 3.11d)} \\
5)\ v_i + F_i(L_x,y,H_x) + F_i(L_x,W_x-w_i-y,H_x) \\
\qquad + F_i(x,w_i,H_x) + F_i(L_x-l_i-x,w_i,H_x) \\
\qquad + F_i(l_i,w_i,H_x-h_i-z) + F_i(l_i,w_i,z) \\
\quad \text{(see Fig. 3.11e)} \\
6)\ v_i + F_i(L_x,y,H_x) + F_i(L_x,W_x-w_i-y,H_x) \\
\qquad + F_i(x,w_i,h_i) + F_i(L_x-l_i-x,w_i,h_i) \\
\qquad + F_i(L_x,w_i,H_x-h_i-z) + F_i(L_x,w_i,z) \\
\quad \text{(see Fig. 3.11f)} \\
7)\ F_{i-1}(L_x,W_x,H_x)
\end{cases}
$$

Let $\underline{e}_i$ be the unit vector. The ith element of $\underline{e}_i$ is set to one, and all other elements are set to zero. The corresponding allocation vectors $\underline{a}_i(L_x,W_x,H_x)$ for above seven possible function values are:

$$
\begin{aligned}
1)\ & \underline{e}_i + \underline{a}_i(L_x,W_x,z) + \underline{a}_i(L_x,W_x,H_x-h_i-z) \\
& + \underline{a}_i(x,W_x,h_i) + \underline{a}_i(L_x-l_i-x,W_x,h_i) \\
& + \underline{a}_i(l_i,y,h_i) + \underline{a}_i(l_i,W_x-w_i-y,h_i)
\end{aligned}
$$

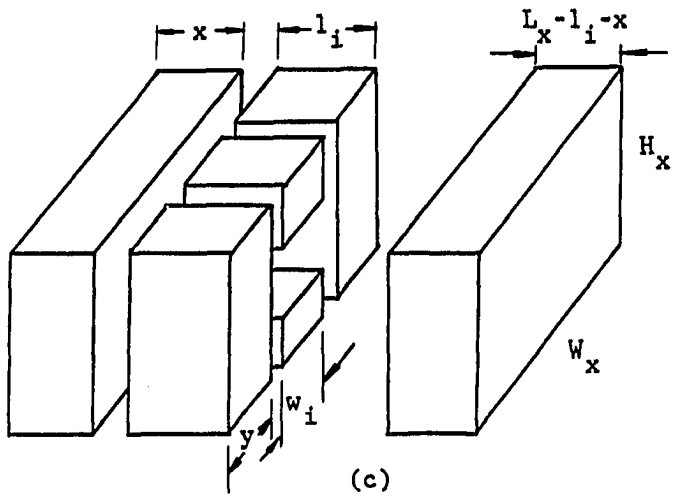


(a)



(b)

Figure 3.11. Six combinations of sub-index pallets



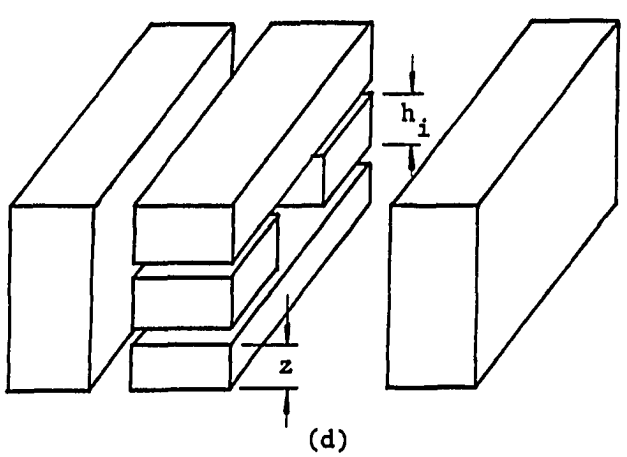
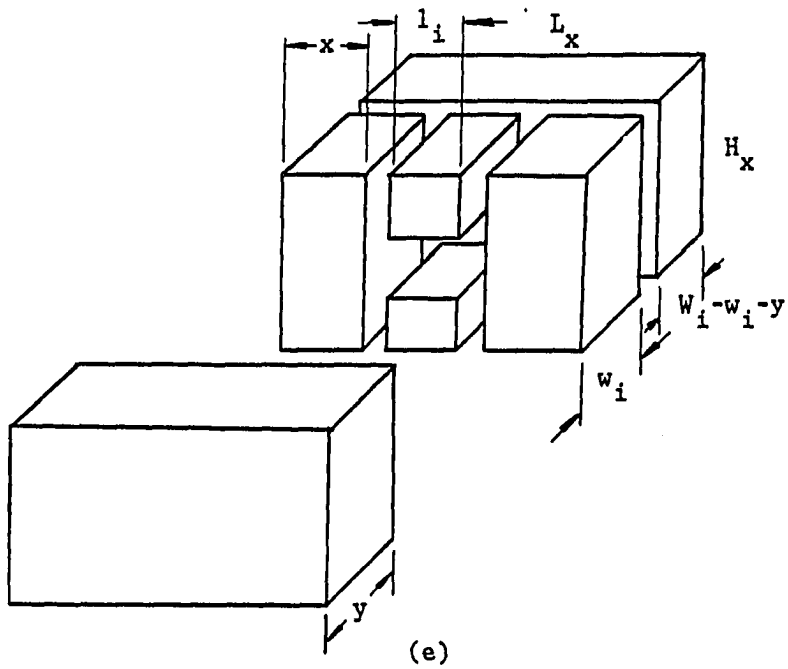
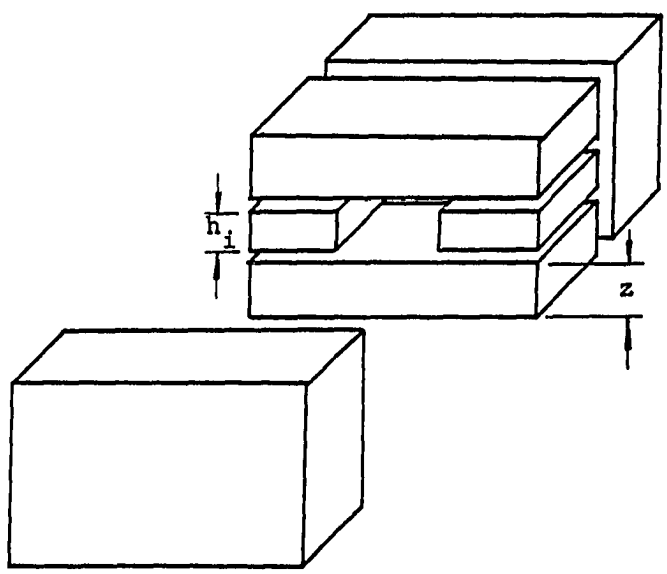


Figure 3.11. continued



(e)



(f)

Figure 3.11. continued

2) through 6) can be obtained using a calculation similar to that shown in 1)

7) $\underline{a}_{i-1}(L_x,W_x,H_x)$

c. The DP procedure Based on the three-dimensional allocation algorithm described above, the following heuristic dynamic programming procedure applies.

STEP 1: Reorder the box types $v_1, v_2, \ldots, v_r$ such that

$$v_1 \leq v_2 \leq \ldots\ldots \leq v_r.$$

STEP 2: Select the box type v_1 to be allocated at the first stage. Let [t] represent the largest integer number no greater than t. Compute $\underline{a}_1(L_x,W_x,H_x) = [L_x/l_1]\cdot[W_x,/w_1]\cdot[H_x/h_1]$, $F_1(L_x,W_x,H_x) = \underline{a}_1(L_x,W_x,H_x)\cdot v_1$, for all L_x, W_x, H_x. Let $i = 2$.

STEP 3: Compute the best return function value and allocation vector for every index pallet (L_x,W_x,H_x).

(a) For $L_x = 1,2,\ldots,L$;

$W_x = 1,2,\ldots,W$;

$H_x = 1,2,\ldots,H$;

allocate a box v_i at coordinate (x,y,z) in the index pallet (L_x,W_x,H_x). Let MAX = 0, and

$AL = \min(L_x-l_i,[L_x/2])$;

$AW = \min(W_x-w_i,[W_x/2])$;

$AH = \min(H_x-h_i,[H_x/2])$.

(b) For $x = 1,2,\ldots,AL$;

$y = 1,2,\ldots,AW$;

$z = 1,2,\ldots,AH$;

calculate the function value $F_i(L_x,W_x,H_x)$ using eq. (3.63) and the associated allocation vector $\underline{a}_i(L_x,W_x,H_x)$.

(c) If MAX < $F_i(L_x,W_x,H_x)$, then

$$MAX = F_i(L_x,W_x,H_x) \text{ and}$$

$$\underline{a}^* = \underline{a}_i(L_x,W_x,H_x).$$

If (x,y,z) is not increased up to (AL,AW,AH), go to (b). Otherwise, go to (d).

(d) Let $F_i(L_x,W_x,H_x)$ = MAX, and

$\underline{a}_i(L_x,W_x,H_x) = \underline{a}^*$

If (L_x,W_x,H_x) is not increased up to (L,W,H), go to (a). Otherwise, go to STEP 4.

STEP 4: Let i = i + 1

If $i \leq r$ (total number of box types),

go to STEP 3. Otherwise, go to STEP 5.

STEP 5: Deliver the solutions $F_r(L,W,H)$ and $\underline{a}_r(L,W,H)$.

2. Restriction on the number of boxes (goal 2)

The dynamic programming procedure for goal 1 as described in the last section only tends to maximize the utilization of a pallet cube without placing any restriction on the number of boxes of each type to be loaded. To make the box proportion of each type satisfy some user-specified number, criteria have been developed to select a rational solution from the two conflicting goals. The criteria are determined based on the number of boxes that have been allocated in the index pallet currently considered.

Basically, the procedure for goal 2 selects a return function whose associated allocation vector best fits the criteria. In case that no allocation vectors satisfy the criteria, the dynamic programming procedure then selects the return function that maximizes the allocated space of a given index pallet as though no restriction is placed on the number of boxes of each type.

The selection criteria for a best allocation vector are based on the value of box ratio which is defined as follows:

Denote r = total number of box types to be allocated

R_g = the desired box proportion of type g as defined previously, $g = 1,2,\ldots,r$

n_g = the desired number of boxes of type g, which is determined using eq. (3.43), i.e.,

$$n_g = \frac{R_g \cdot V}{\sum_{i=1}^{r} R_i \cdot v_i}$$

where V is the pallet's volume;

v_i is the box's volume of type i.

a_{ij} = the jth element (number of type j boxes allocated) of the allocation vector $\underline{a}_i(L_x, W_x, H_x)$, $i = 1,2,\ldots,r$, $j = 1,2,\ldots,i$.

BR = box ratio

$$= \sum_{j=1}^{r} \frac{a_{ij}}{n_j}, \text{ for stage } 1,2,\ldots,r-1 \qquad (3.64)$$

$$= \sum_{j=1}^{r} \text{ABS}\left(\frac{a_{ij}}{\sum_{k=1}^{r} a_{ik}} - R_j\right), \text{ for stage } r \qquad (3.65)$$

where ABS(·) returns the absolute value;

$$\frac{a_{ij}}{\sum_{k=1}^{r} a_{ik}} = \text{box proportion of type } j \text{ in the allocation vector } \underline{a}_i(L_x, W_x, H_x).$$

In each stage of the dynamic programming procedure, the box ratio should be as small as possible. The box ratio of eq. (3.64) calculates the cumulative proportions of the allocated number to the desired number of boxes. The allocation vector associated with the minimum box ratio, eq. (3.64), tends to have the allocated number far smaller than the desired number for each type of box. This allows the subsequent stages to have more box allocation alternatives. The box ratio of eq. (3.64) is calculated for all allocation vectors in all stages except the last stage (stage r) of the dynamic programming procedure.

The box ratio of eq. (3.65) calculates the cumulative difference between the box proportion of an allocation vector and the desired box proportion specified by the user. The allocation vector associated with the minimum box ratio, eq. (3.65), has the allocated box proportion closest to the desired box proportion. This box ratio is only calculated for the allocation vectors in the last stage (the last box size to be allocated). Recall that each stage of the DP procedure considers only one box size of a unique orientation. In case that the box types to be allocated in the last two stages have the same dimensions with different orientations (l-by-w and w-by-l), box ratio of eq. (3.65) should be applied for the last two stages of the DP procedure.

The following four rules describe the decision criteria used for selecting a best function value and allocation vector between goals 1 and 2.

Rule 1: For a given index pallet (L_x,W_x,H_x) and the placement location (x,y,z), compute the box ratio for each allocation vector of the six combinations of sub-index pallets (see Figures 3.11a-f), and the allocation vector at the previous stage. In case that two or more combinations have the same best function values, select the allocation vector that has the minimum box ratio. In this way, the subsequent process will have more box allocation alternatives so as to obtain a maximum return function value.

Rule 2: For a given index pallet (L_x,W_x,H_x), suppose that the maximum function values are the same for some different box locations (x,y,z), where

$$x = 1,2,\ldots, \min(L_x-l_i,[L_x/2]),$$

$$y = 1,2,\ldots, \min(W_x-w_i,[W_x/2]),$$

$$z = 1,2,\ldots, \min(H_x-h_i,[H_x/2]),$$

then select the allocation vector such that its box ratio is as small as possible.

Rule 3: For a given index pallet (L_x,W_x,H_x), suppose no allocation vector $\underline{a}_i(L_x,W_x,H_x)$ exists such that

$$a_{ij} \leq n_j, \text{ for all } j,$$

then let $F_i(L_x,W_x,H_x)$ = maximum function value without considering the

restriction on the number of boxes. Since no solution exists for goal 2, the return function that has the maximum value is selected. Rules 1 and 2 are still applied to select the maximum value.

Rule 4: For a given index pallet, suppose that at least one allocation vector $\underline{a}_i(L_x,W_x,H_x)$ exists such that

$$a_{ij} \leq n_j, \quad \text{for all } j,$$

then let $F_i(L_x,W_x,H_x)$ = the best function value selected from those that their corresponding allocation vectors satisfy $a_{ij} \leq n_j$, for all j. In this case, the maximum function value and its corresponding allocation vector are selected among the candidates which have all their allocated number less than the desired number of boxes of every type.

The above four selection rules for determining a best return function value and allocation vector are implemented in STEP 3 of the dynamic programming procedure for goal 1 (see the last section).

A BASIC program implementing the heuristic dynamic programming procedure is presented in Appendix B. This program has been employed to determine the optimal pallet patterns used in the simulation.

c. Post-solution adjustment Because of the design nature of the heuristic dynamic programming procedure, goal 1 (maximizing the utilization of a pallet cube) has a higher priority than goal 2 (satisfying desired box proportion). The final solution generated by the dynamic programming procedure may only give a value that approximates the desired box proportion.

However, a larger box can be always substituted with one or more smaller boxes and still maintain the feasibility of the pallet pattern. A feasible pallet pattern also holds by deleting some number of boxes from the heuristic DP solution, if the stability of a pallet load is not the main consideration. Many alternatives with different box proportions and pallet space utilizations can be generated by substituting or reducing number of boxes. It is up to the user to weigh the importance between goals 1 and 2, and select the best pallet pattern from these alternatives.

Consider the following example. Boxes of two different sizes, 1x2x1 and 2x2x2, are to be loaded onto a pallet of size 4x4 with the stacking height of 4. The desired box proportions for box sizes 1x2x1 and 2x2x2 are 0.67 and 0.33, respectively. The heuristic dynamic programming procedure yields the following solution to this problem.

8 boxes for 1x2x1, and

6 boxes for 2x2x2.

This solution completely fills the pallet space of 64. The solution's corresponding box proportions are 0.57 and 0.43 for box sizes 1x2x1 and 2x2x2, respectively, which are not the desired box proportions.

Nevertheless, one box of 2x2x2 is equivalent to four boxes of 1x2x1. With this substitution, the original solution can be changed to

12 boxes for 1x2x1, and

5 boxes for 2x2x2.

This solution still maximizes the use of pallet space. The solution's corresponding box proportions are 0.70 and 0.30 for box sizes 1x2x1 and 2x2x2, respectively. These also differ from the desired box proportions.

By deleting two boxes of size 1x2x1, the above solution can be further changed to

10 boxes for 1x2x1, and

5 boxes for 2x2x2.

This combination of boxes uses only 60/64's of the pallet's volume. However, this solution gives exactly the desired box proportions, which are 0.67 and 0.33 for box sizes 1x2x1 and 2x2x2, respectively. The solution revisions are summarized in Table 3.3.

The user may select the solution of box substitution if he/she feels the use of pallet space is more important than box proportion. In contrast, other solutions may be selected if the desired box proportion must be satisfied.

Table 3.3. Solutions with substitution and reduction of boxes

Solution	Number of box		Box proportion		Use of pallet space
	(1x2x1)	(2x2x2)	(1x2x1)	(2x2x2)	
Heuristic DP	8	6	0.57	0.43	64
Box substitution	12	5	0.71	0.29	64
Box reduction	10	5	0.67	0.33	60

The heuristic dynamic programming procedure can be used to determine the substituted number of smaller boxes for one larger box. The base of the larger box can be considered as a pallet. The length and the width of the larger box are used as the dimensions of the pallet. The height of the larger box is used as the stacking height of the pallet. One or more smaller box sizes can be considered in the dynamic programming procedure to substitute for the larger box.

With the substitution and reduction of boxes, many possible alternate pallet patterns can be created. The user may select the "best" pallet pattern from these alternatives according to the importance of box proportion and pallet space utilization.

The developed mixed 0-1 integer programming and heuristic dynamic programming models can be used to generate an optimal pallet pattern. The placement location data associated with the pallet pattern can be used as input to a robotic control program for automatic palletizing.

The development of robotic palletizing programs and design of the robotic palletizing system are the subjects of the following chapter.

IV. PALLETIZING SYSTEMS AND ROBOT PROGRAMMING

A. Introduction

Robotic palletizing systems can be implemented for both warehousing and manufacturing industries. For warehousing, goods are collected from a reserve storage area according to customer orders. For manufacturing industries, products are usually produced according to master production schedules. Robotic palletization can be most cost effective for the factories producing many products in small lots and in different sizes. Robots provide the best solution to accommodate the continuous changeover of products since many pallet patterns can be simultaneously stored in computer memory.

Because of the random nature of incoming box sizes to the robotic palletizing cell, care must be taken in the overall robotic palletization design. Two palletizing approaches -- dynamic pallet patterns, and multi-pallet packing with turntables -- can be used to overcome the random arrivals of various box sizes. These two approaches can reduce the demands on off-line box storage areas and robot movements to and from these storage areas.

A Rhino XR-2 robot [83] has been employed in this research to investigate the performance and feasibility of the robotic palletizing system. The hardware and software aspects used to construct a robotic palletizing cell are presented in this chapter. An algorithm that transforms the x, y, z coordinates to Rhino robot's joint coordinates

has been developed. Since the repeatability of the Rhino XR-2 robot is not sufficient for this palletizing task, a software control program has also been developed to enable the Rhino robot to reset itself to its home position.

In addition, a palletizing control program that carries out the entire robotic palletizing operations is described in this chapter. Data input requirements for the palletizing program are also presented.

B. Two Palletizing Approaches

1. Dynamic pallet patterns

Boxes of various sizes arrive at the robotic palletizing cell in a random fashion. Whenever box size distributions (proportions) from the in-feeding conveyor are significantly changed, a new pallet pattern should be immediately generated to accommodate this variation. However, due to long computation time requirements and limited computation capability of present industrial robots, on-line determination of an optimal pallet pattern using the developed mixed 0-1 integer or heuristic dynamic programming model is not feasible.

One method to cope with this problem is to employ a "jukebox" approach. By taking advantage of large memory storage capability of industrial robots, ten or more different pallet patterns may be simultaneously stored in computer memory. The possible pallet patterns can be pre-determined using the formulated models with different mixes of box size proportions. When a box size distribution is changed, the computer then searches for an appropriate pallet pattern whose pre-

determined box size proportions represent the best "match" of the proportions of box sizes to be loaded. In this way, more boxes can be directly loaded onto pallets without going through the off-line storage areas.

Automatic monitoring devices connected with the robot's control unit can be employed to detect the change of box size distributions. Every box on the in-feeding conveyor is recorded before it arrives at the robotic palletizing cell.

When a pallet is full, the computer examines the boxes in queue to determine the incoming box size distribution. The look-ahead queue length of boxes can be determined using the following expression.

$$\sum_{i=1}^{Q-1} v_i < \alpha V \leq \sum_{i=1}^{Q} v_i \qquad (4.1)$$

where v_i = the volume of the ith box in queue; the first box, v_1, is the one at the robot's pick-up position.

V = the volume of a pallet

α = look-ahead factor of queue length

Q = the last box of the look-ahead queue

= total number of boxes in the look-ahead queue

Initially, the look-ahead factor α is set to 1, i.e., boxes in queue are counted until the cumulative box volumes reach the volume of a pallet. The frequency of each box size in the look-ahead queue is converted to a proportion, which is calculated as follows:

$$\text{box proportion of size } i = \frac{\text{number of boxes of size } i \text{ in observed queue length}}{\text{total number of boxes, } Q}$$

The calculated box size proportions are compared with the reference proportions which are stored in computer memory prior to the palletizing process. If the difference between the calculated and reference proportions exceeds some user-specified tolerance, the robot's control units will search for an appropriate pallet pattern whose reference proportions match fairly well with the box size proportions of the observed queue. The robot's control unit will then signal the robot and switch the pallet pattern.

In case that the calculated box size proportions determined from eq. (4.1) do not yield a good match with any pre-stored reference proportions, the look-ahead factor α may be altered to a larger number, say 2. This extends the look-ahead queue length. The maximum value of α may be subject to the conveyor length and available space of the off-line storage areas.

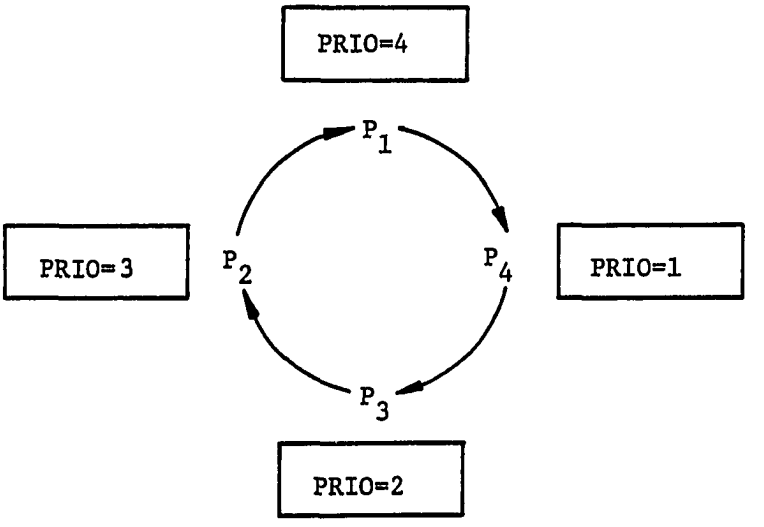
The determination of a best look-ahead factor α will be carried out using simulation in a latter section of Chapter V.

2. Multi-pallet packing with turntables

With multi-pallet packing, the robot can load two or more pallets simultaneously so long as the robot's work envelope is sufficiently large. Whenever a box is picked up by the robot, the palletizing control program will search for an available pallet space from these simultaneously loaded pallets. In case that the placement of the box

on any one of the pallets is not possible, the robot places it in the off-line storage area. The stored box will be loaded onto the pallet as soon as a pallet space for the specific box size becomes available.

For multi-pallet packing, a priority number is assigned to each pallet. The pallet with the highest priority number receives a required box first. When a pallet is fully loaded, it will be removed using fork-lift trucks or automatic transfers. An empty pallet is then inserted. A lowest priority number is reassigned to this empty pallet. The priority numbers are increased for the remaining pallets which are not fully loaded. Consider a four-pallet packing operation. The change of priority number of the four pallets can be represented by a rotation cycle as shown below.



where P_i = pallet number i

PRIO = priority number; the larger the number, the higher the priority

Initially, assign priority 4 to pallet 1,
priority 3 to pallet 2,
priority 2 to pallet 3, and
priority 1 to pallet 4.

When pallet 1 is full, then reassign
priority 4 to pallet 2,
priority 3 to pallet 3,
priority 2 to pallet 4, and
priority 1 to pallet 1.

This is equivalent to rotating clockwise the cycle by 90°. Likewise, the new assigned priority numbers can be obtained in this manner whenever a pallet is full.

Due to the limitations on the size of the robot's work envelope, at most three 48" by 40" GMA (Grocery Manufacturers of America) or 1200 mm by 800 mm (European Exchange) pallets can be simultaneously loaded.

One way to increase the robot's work envelope is to use moving-base tracks. With this method, the robot is mounted on some form of transport system which moves along a set of tracks. This method requires the installation of the transport system which may not be possible or economical. Also, an industrial robot can be as heavy as 5,000 lbs. This means a powerful drive system is required for transport devices, and a relatively long traveling time results.

To minimize the cycle time of loading a box onto the pallet, turntables can be used instead of moving-base tracks. A turntable can be installed in front of a palletizing robot. Only part of the turntable

needs to be contained within the robot's work envelope. A layout of a four-pallet turntable system is illustrated in Figure 4.1.

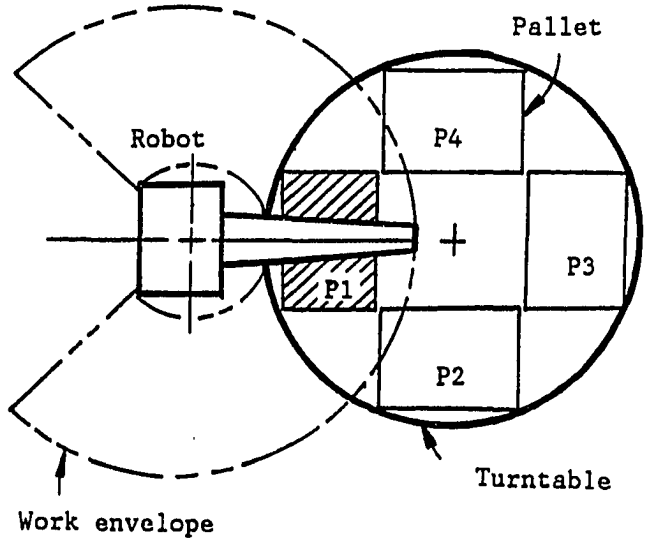


Figure 4.1. A four-pallet turntable system

As many as four pallets can be simultaneously loaded using this turntable. Note that only pallet 1 (P1 in Figure 4.1) is contained within the work envelope. The turntable interfaces with the robot's control unit. When pallet 2 (P2) is called for, the turntable rotates 90° so that pallet 2 becomes reachable to the robot. The remaining area of the robot's work envelope can be used to install a second turntable, or store boxes not immediately available on the pallet. The four-pallet turntable may be replaced by a larger eight-pallet turntable. The robotic palletizing cell with an eight-pallet turntable is shown in Figure 4.2. This gives each box size at least eight storage buffers.

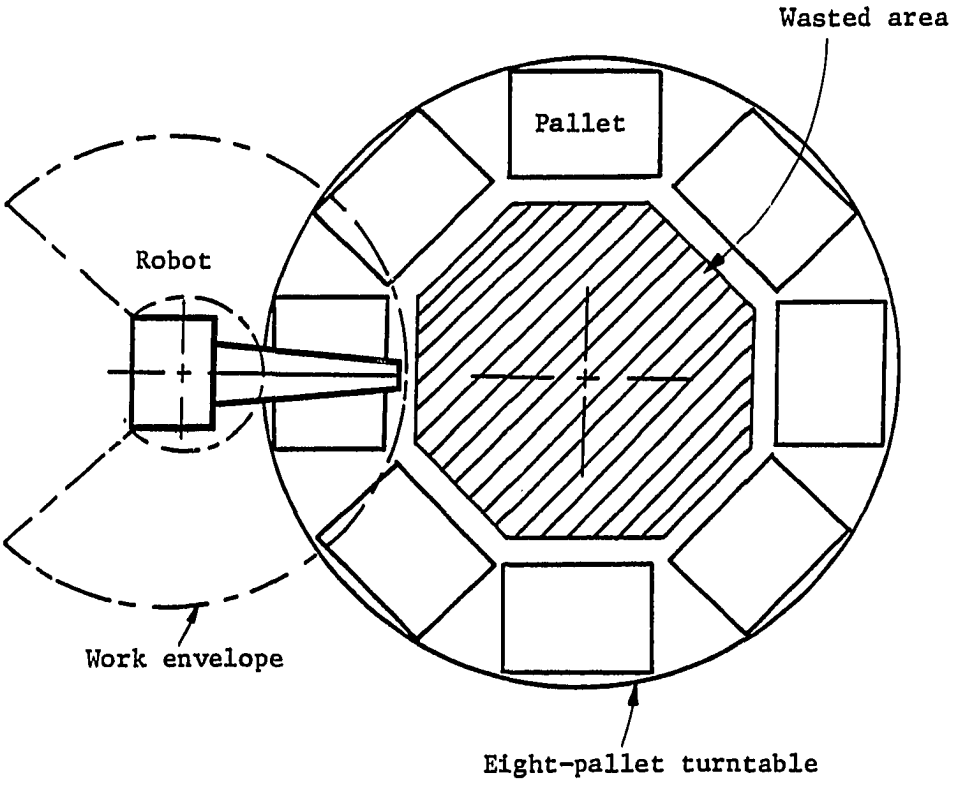


Figure 4.2. An eight-pallet turntable system

However, an eight-pallet turntable results in more wasted space when compared with a four-pallet turntable. The shaded area in Figure 4.2 represents the wasted space of an eight-pallet turntable.

The turntable system also provides flexibility for distribution warehouses. It is especially suited for the warehouses in which many customer orders for small quantities are requested daily. With a four-pallet turntable, four different pallet patterns may be presented

on the turntable simultaneously. Hence, a robotic palletizing station can manage four different customer orders.

For the overall design of a robotic palletizing cell, the dynamic pallet patterns and turntable systems should be integrated together to construct a robust automatic palletization system. The dynamic pallet patterns and multi-pallet packing with a four-pallet turntable have been carried out using a physical simulator. The performance of these two palletizing approaches will be discussed in Chapter V.

C. The Robotic Palletizing Cell

In an industrial palletizing system, the robot used for palletizing must be able to handle boxes with large size and weight ranges. Cartesian, spherical, and jointed-arm industrial robots have been applied successfully in palletizing [80,89]. For instance, a Cincinnati Milacron T^3-566 jointed-arm robot has been successfully used in palletizing [89].

This research has utilized a table-top Rhino XR-2 robot to implement the palletizing task. A Rhino conveyor and a Rhino turntable have also been employed to construct a miniature physical simulator of a robotic palletizing cell. This physical simulator has been used to collect palletizing statistics and verify the performance of the robotic palletizing system.

The palletizing hardware and software requirements using the Rhino XR-2 robot are the subjects of the following sections.

1. The equipment/hardware

a. Rhino XR-2 robot The Rhino XR-2 is a table-top, jointed-arm robot with five degrees of freedom. The movement of the manipulator is driven by the five DC servo motors. Each motor has two optical encoders. One is used to determine how far a motor has moved by counting the encoder holes. The other is used to determine in what direction the motor is moving. The structure of the Rhino XR-2 robot is illustrated in Figure 4.3.

The movements of the five DC motors for base, shoulder, elbow, pitch and roll are controlled via a Rhino controller. The controller

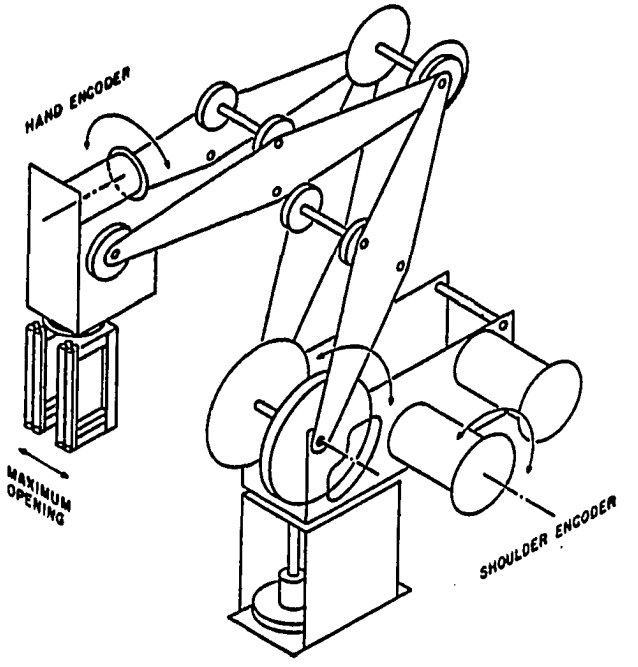


Figure 4.3. The structure of the Rhino XR-2
Source: reference [86]

is an eight-axis controller that is operated from the RS-232C port of a microcomputer. It can control up to eight motors simultaneously.

b. The Rhino conveyor The Rhino conveyor is used as an in-feeding device to deliver boxes into the robot's pick-up position. This conveyor is driven by a DC servo motor and is monitored with an optical encoding system. The conveyor can be directly operated from the Rhino controller under the control of a microcomputer. The Rhino conveyor is shown in Figure 4.4.

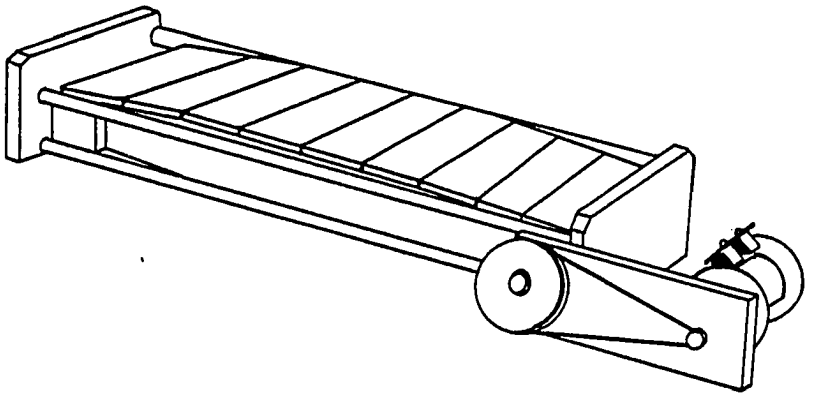


Figure 4.4. The Rhino conveyor

c. The Rhino turntable A Rhino turntable has been employed to simulate the system of multi-pallet packing. The turntable is twelve (12) inches in diameter. Up to four 4" by 4" pallets can be placed on the table at a time. The turntable is also driven by a DC servo motor and is monitored with an optical encoding system. The

Rhino controller supplies the motor with power and provides the decoding for the optical encoders. The Rhino turntable is illustrated in Figure 4.5.

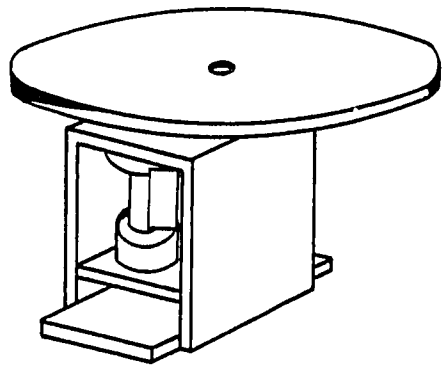


Figure 4.5. The Rhino turntable

d. The Rhino vacuum gripper The Rhino vacuum gripper is used to simulate an industrial vacuum system. It consists of a vacuum cup to allow the lifting of objects by converting the air pressure into vacuum. The actuation and release of holding force of the vacuum cup can be controlled from the Rhino controller under the control of a microcomputer. Since the degrees of repeatability and placement resolution of the Rhino XR-2 robot are not sufficient for the palletizing task, a spring mechanism has been used to accommodate variations in placement accuracy. The configuration of the vacuum gripper and spring yoke is shown in Figure 4.6.

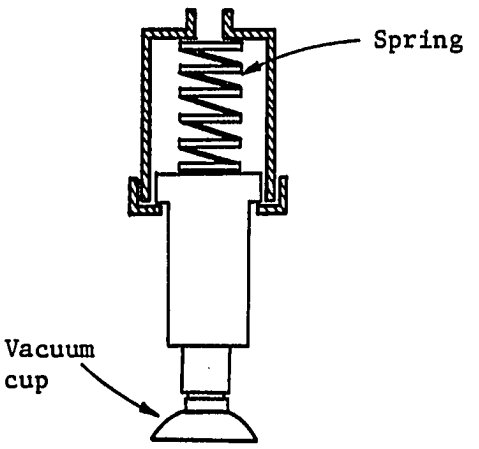
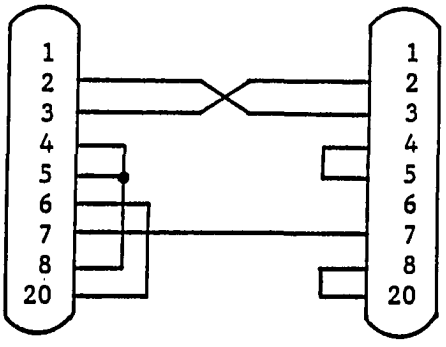


Figure 4.6. The vacuum gripper and spring-loaded yoke

e. The RS-232C interface RS-232C was originally developed to standardize the interface between a modem and a terminal. The Rhino controller (modem) is connected with a T1 Professional computer (terminal) via the RS-232C interface. This serial I/O port transmits each data byte bit by bit between the controller and the computer. The RS-232C interface between the controller and the T1 computer is connected as a null modem. The connections are shown in Figure 4.7. This serial communications protocol uses 9600 baud, 7 bits, even parity and 2 stop bits to transmit data between the controller and the computer. The BASIC statement required to support RS-232C asynchronous communication is presented below.

OPEN "COM1: 9600,E,7,2" AS #<file number>



Host computer Rhino controller

pin 2 : Transmitted Data
pin 3 : Received Data
pin 4 : Request to Send
pin 5 : Clear to Send
pin 6 : Data Set Ready
pin 7 : Signal Ground
pin 8 : Data Carrier Detect
pin 20 : Data Terminal Ready

Figure 4.7. The connections between computer and controller

2. Software for the Rhino robot

a. Rhino control commands The T1 Professional computer can give the Rhino controller four separate BASIC commands to control eight DC motors. These four commands are:

- PRINT #<filename>,"abnnn" : STARTS a specified motor with a specified number of encoder holes in a specific direction. The specified number is added to the error register. In

this PRINT statement,

a = A,B,C,D,E,F,G, or H corresponds to one of eight motors;

b = + or - represents the motor's moving direction;

nnn = a three-digit number ranging from 000 up to 127.

- PRINT #<filename>,"aX" : STOPS a specified motor instantly and clears the rror register. In this PRINT statement,

  a = A,B,C,D,E,F,G, or H corresponds to one of eight motors.

- PRINT #<filename>,"a?" : READS the error register and returns the value indicating how far a specified motor is from its stopping point. In this PRINT statement,

  a = A,B,C,D,E,F,G, or H corresponds to one of eight motors.

- PRINT #<filename>,"I" : requests the controller for the STATUS of six interrupt lines. The six lines are organized as the limit switches for motors C, D, E, F, G, and H. They are used to detect the limit switch closures for these six motors. This status command can be used to reset the Rhino robot to its hard home position. If all microswitches are open, the controller will return a number of 95. If any combination of the microswitches is closed, this number will be decreased as follows:

Switch	Decrement value
C	1
D	2
E	4
F	8
G	16
H	32

b. Coordinate transformations There are two approaches to control the movements of the manipulator. One is a teach program. The other is a Cartesian coordinate program. With the teach program, the user enters the data manually by moving the arm to each of the desired positions. The computer counts and memorizes the number of encoder holes each joint moves.

The Cartesian coordinate program accepts information of the locations which are defined by their Cartesian coordinates. The x, y, z coordinates of desired positions are transformed to the robot's joint coordinates. The joint angles of the robot arm are then converted to the encoder holes. These transformed values can be used by the control program to control each activity of the joint.

Due to the complexity of palletizing operations, and the high number of possible pick-and-place positions, the teach program is often not practical. A Cartesian coordinate program has been developed to control the Rhino robot for palletization applications. This section described how Cartesian and joint coordinates are defined and how the transformation solutions are derived. This transformation solution is based on the procedure described in reference [79].

The kinematic model of the Rhino XR-2 robot is shown in Figure 4.8. The model indicates how each joint is articulated, how the joint angles are measured, and the distance between joints.

Define θ_1 = the joint angle of the base,

θ_2 = the joint angle of the shoulder,

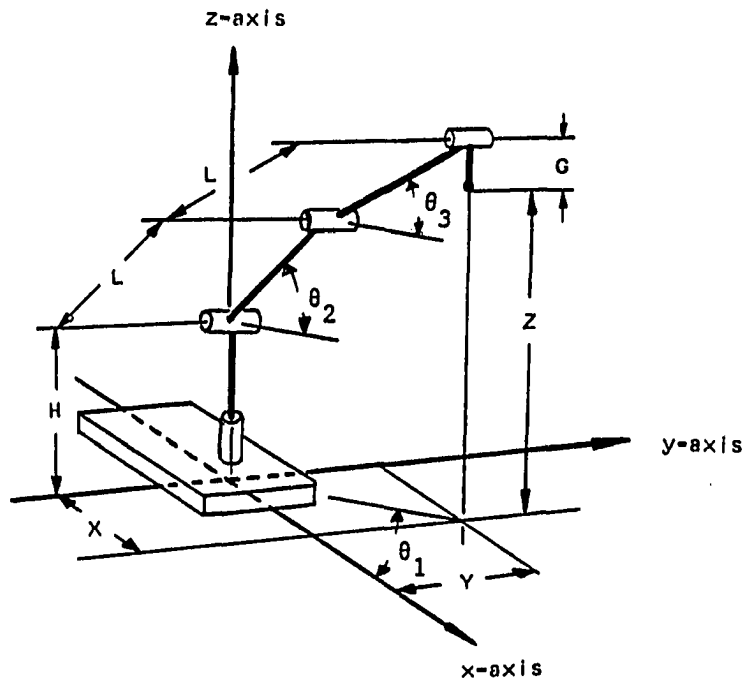


Figure 4.8. Kinematic model of the Rhino XR-2

θ_3 = the joint angle of the elbow,

H = distance from the table-top to the shoulder joint,

L = distance from shoulder joint to the elbow joint,

= distance from elbow joint to the wrist joint,

G = distance from wrist joint to the end of the gripper,

X = the distance of the desired end point in front of the arm; measured from the base pivot along the x-axis,

Y = the distance of the desired end point to the left (or right) of the arm; measured from the base pivot along the y-axis,

Z = the vertical height of the desired end point above the table-top.

These symbols are also shown in Figure 4.8.

It is desired to keep the gripper always parallel to the z-axis. Therefore, no transformation of the wrist angle is required. The first step of the transformation is to determine the base angle, θ_1. The top view of the arm is shown in Figure 4.9.

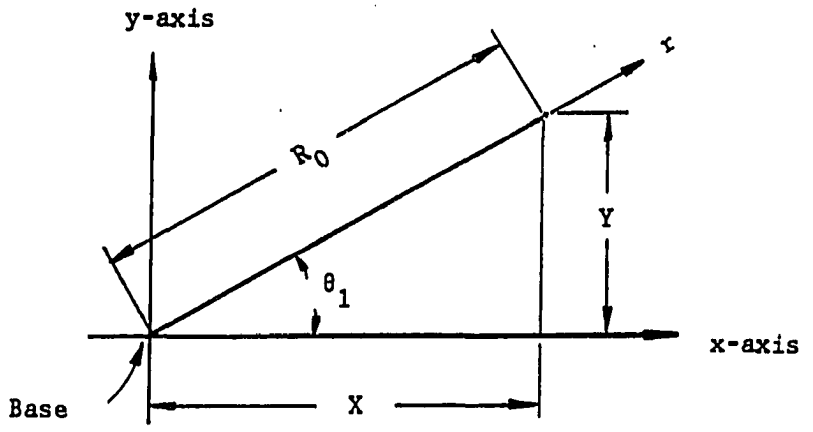


Figure 4.9. Top view of the arm

It follows that

$$\theta_1 = \begin{cases} 90°\text{SGN}(Y), & \text{for } X = 0 \\ \text{TAN}^{-1}(Y/X), & \text{for } X > 0 \\ \{180° - \text{TAN}^{-1}(|Y/X|)\}\text{SGN}(Y), & \text{for } X < 0 \end{cases} \qquad (4.2)$$

where SGN(Y) = 1, if y > 0,

= -1, if y < 0.

The second step is to determine the shoulder angle, θ_2, and the elbow angle, θ_3. The shoulder-elbow-wrist triangle is shown in Figure 4.10 using the new translated coordinate system in which the r-axis is the one along the direction of the robot arm (see Figure 4.9). The new origin (0,0) is defined to be at the shoulder joint.

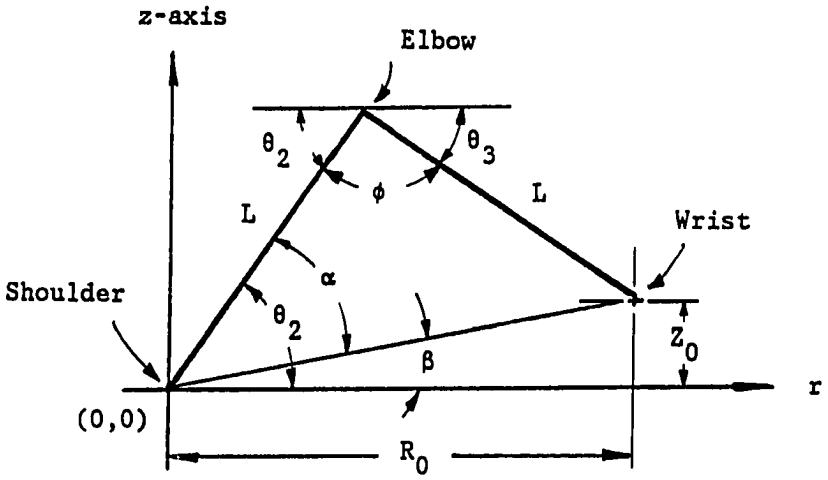


Figure 4.10. The shoulder-elbow-wrist triangle

$$\text{Let } R_0 = \sqrt{X^2 + Y^2}\,, \tag{4.3}$$

$$Z_0 = Z + G - H, \tag{4.4}$$

as defined in Figures 4.9 and 4.10.

Three new angles α, β, and ϕ defined in Figure 4.10 are introduced to obtain the solution. From Figure 4.10, obtain

$$\beta = \text{TAN}^{-1}(Z_0/R_0) \tag{4.5}$$

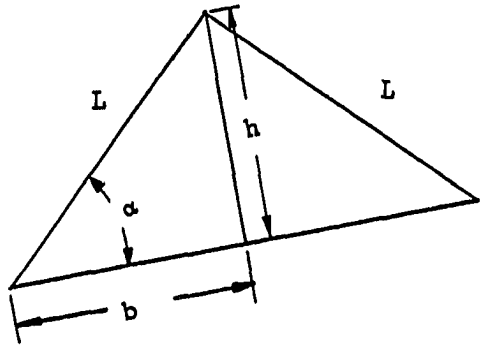


Figure 4.11. Simplified triangle

The shoulder-elbow-wrist triangle is now redrawn as shown in Figure 4.11. The triangle can be partitioned into two equivalent right triangles. The length of each base, b, and the height, h, of the right triangles are

$$b = (\sqrt{Z_0^2 + R_0^2})/2 ,$$

and

$$h = \sqrt{L^2 - b^2} .$$

This yields

$$\alpha = \text{TAN}^{-1}(h/b) = \text{TAN}^{-1}\sqrt{4L^2/(R_0^2 + Z_0^2) - 1} . \qquad (4.6)$$

From Figure 4.10,

$$\theta_2 = \alpha + \beta \qquad (4.7)$$

Since $\theta_2 + \phi + \theta_3 = 180°$ and $\phi + \alpha + \alpha = 180°$, then

$$\theta_3 = \alpha - \beta$$

The elbow angle θ_3 is defined as the angle above the horizontal. The sign of θ_3 should be changed since θ_3 in Figure 4.10 is measured below the horizontal. Therefore,

$$\theta_3 = \beta - \alpha. \tag{4.8}$$

The third step of the coordinate transformation is to determine the roll angle of the hand, θ_4. In this research, the hand always points straight down. Also, the pallets are parallel to the x- or the y-axis when they are placed in the work envelope. Therefore, the Rhino robot hand must be parallel to the y-axis no matter where the arm moves.

Roll should be measured with respect to the Cartesian frame. With the hand pointing straight down, the roll angle to keep the hand orientation fixed along the y-axis can be determined from the base angle, θ_1. The roll angle in the four quadrants of the x-y coordinate is calculated as follows (see Figures 4.12 and 4.13):

Let $|\theta|$ represent the absolute value of the angle θ.
When $X \geq 0$,

In quadrant I, the roll magnitude = $|\theta_1|$ and the wrist should rotate clockwise (position direction).

In quadrant II, the roll magnitude = $|\theta_1|$ and the wrist should rotate counterclockwise (negative direction).

Thus, the roll angle $\theta_4 = \theta_1$, for $X \geq 0$.
When $X < 0$,

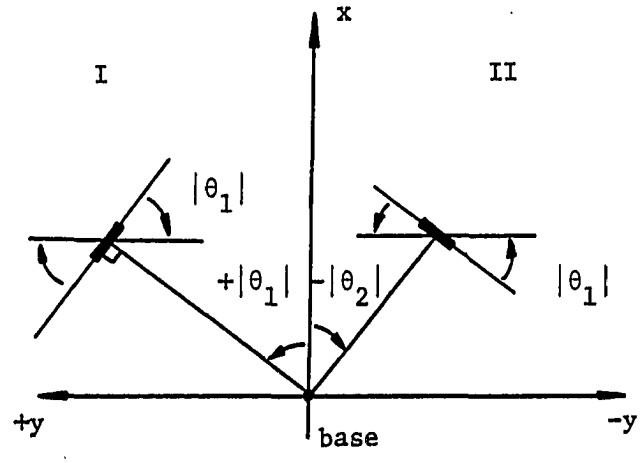


Figure 4.12. Roll angles in quadrants I and II

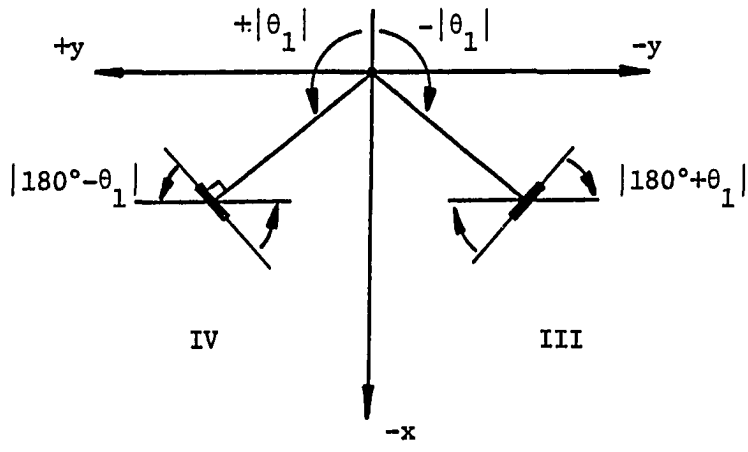


Figure 4.13. Roll angles in quadrants III and IV

In quadrant III, the base angle, θ_1, is a negative number.

The roll magnitude = $|180° + \theta_1|$ and the wrist should rotate clockwise.

In quadrant IV, the base angle, θ_1, is a positive number.

The roll magnitude = $|180° - \theta_1|$ and the wrist should rotate counter-clockwise.

Thus, the roll angle $\theta_4 = 180° + \theta_1 > 0$, for $X < 0$ and $Y \leq 0$;

$= \theta_1 - 180° < 0$, for $X < 0$ and $Y > 0$.

The last step of the coordinate transformation procedure is to convert the joint angles to the encoder holes. The conversion factors between encoder holes and joint angles are shown in Table 4.1.

Table 4.1. Conversion factors between holes and joint angles (source: Reference [83])

Motor	Joint	Holes per revolution	Holes per degree
A	Roll	1496.0	4.15
B	--	1496.0	4.15
C	Gripper	4541.3	12.61
D	Elbow	3144.0	8.73
E	Shoulder	3144.0	8.73
F	Base	2620.0	7.28
G	Conveyor	2620.0	7.28
H	Turntable	2620.0	7.28

A summary of this coordinate transformation solution is given in Table 4.2. A BASIC program implementing this solution is presented in Appendix C (subroutine starting at line 4230).

c. Simultaneous movement of the joints Since the Rhino's START command (PRINT "abnnn" described previously) can only move one motor/

Table 4.2. Summary of the coordinate transformation

Step	Operation
1	Determine X, Y, Z
2	Base angle: $\theta_1 = \begin{cases} 90°*SGN(Y), & \text{for } X = 0 \\ TAN^{-1}(Y/X), & \text{for } X > 0 \\ \{180° - TAN^{-1}(\lvert Y/X \rvert)\}*SGN(Y), & \text{for } X < 0 \end{cases}$
3	$R_0 = \sqrt{X^2 + Y^2}$
4	$Z_0 = Z + G - H$
5	$\beta = TAN^{-1}(Z_0/R_0)$
6	$\alpha = TAN^{-1}\sqrt{4L^2/(R_0^2+Z_0^2) - 1}$
7	Shoulder angle: $\theta_2 = \alpha + \beta$
8	Elbow angle: $\theta_3 = \beta - \alpha$
9	Roll angle: $\theta_4 = \begin{cases} \theta_1, & \text{for } X \geq 0 \\ 180° + \theta_1, & \text{for } x < 0 \text{ and } y \leq 0 \\ \theta_1 - 180°, & \text{for } x < 0 \text{ and } y > 0 \end{cases}$
10	Convert joint angles to encoder holes

joint at a time, the robot movements may look awkward if the three major joints (base, shoulder and elbow) are moved separately. The robot's movements between the current position and target position should be a relatively smooth path. This means that if the base joint moves 200

encoder holes, the shoulder joint moves 8 holes, and the elbow joint moves 4 holes, the movements should be broken up so to have smaller movement ratios as shown below.

Movement sequence	Ratio (base : shoulder : elbow)
1	50 : 2 : 1
2	50 : 2 : 1
3	50 : 2 : 1
4	50 : 2 : 1
Total	200 : 8 : 4

The software has to take on the task of interpolating the commands so that the movements of the base, the shoulder and the elbow are at least pseudo-simultaneous. The roll movement of the hand is separated from the three joints. Roll movement is carried out after the three major joints complete their movements.

The error register for each motor can only store a maximum count of 127. Therefore, the encoder holes to be moved must be divided into smaller values to prevent register overflow. The Rhino's read command (PRINT "a?") is used to read out the current stored number in the error register. If the read-out number is less than 127, then a small number may be added to the register. If this is not possible, the program must wait until the number in the register is reduced to an acceptably low number. This procedure of adding small number to the register is repeated until the total numbers added reach the amount of desired encoder holes for a specific motor movement. Furthermore, the

movement in terms of encoder holes for each joint should be as small as possible so that a smooth and simultaneous-like trajectory can be obtained. However, the number of encoder holes for each individual movement should not be too small; otherwise, the time delay and jerks between robot movements may occur. From experiments, it has been found that six is an adequate number to obtain smooth movements of the robot joints.

The procedure to yield smooth and pseudo-simultaneous movements of the base, the shoulder and the elbow is described as follows:

STEP 1: Obtain the numbers of encoder holes of the three major joints using the coordinate transformation program.

STEP 2: Sort the numbers of encoder holes of the three major joints in nonincreasing order.

Let the order be N1, N2 and N3, and N1 $\geq$ N2 $\geq$ N3.

STEP 3: Calculate the movement ratios and initialize the parameters.

Let R2 = N2/N1 and R3 = N3/N1;

```
COUNT = 0;
MAXHOLE = N1;
STORE2 = 0;
STORE3 = 0.
```

STEP 4: Check whether the current number in the error register related to N1 is less than 121. If it is larger than 121, then wait until the number is decreased to 121 or less. Otherwise, move the motor associated with N1 by the amount of encoder holes which is specified by the value of

min(6, MAXHOLE).

Let COUNT = COUNT + min(6, MAXHOLE)

MAXHOLE = MAXHOLE - 6

STEP 5: Let ROTATE = INT(COUNT*R2) and

MOVE = ROTATE - STORE2,

where INT(•) returns a largest integer number no greater than the one in the parentheses.

Move the motor associated with N2 by the amount of holes specified by the value of MOVE.

Let STORE2 = ROTATE.

STEP 6: Let ROTATE = INT(COUNT*R3) and

MOVE = ROTATE - STORE3.

Move the motor associated with N3 by the amount of holes specified by the value of MOVE.

Let STORE3 = ROTATE.

STEP 7: If MAXHOLE is decreased to zero or a negative number, terminate the procedure. Otherwise, go to STEP 4.

A BASIC program implementing this procedure is presented in Appendix C (subroutine starting at line 4490).

d. Self-resetting Since the hardware repeatability of the Rhino XR-2 robot is not sufficient for the palletizing task, a software control program has been developed to overcome this problem. This software program allows the Rhino robot to reset itself to its "hard" home position.

In the Rhino XR-2 system, the reset position (hard home position) is defined as follows:

- Robot facing straight forward on the base (along the x-axis)
- Shoulder straight up (along the z-axis)
- Elbow straight out forward (parallel to the x-axis)
- Hand straight down (parallel to the z-axis)

This hard home position corresponds to the x, y, z coordinates, 9", 0", 12.8", respectively.

The Rhino's hand position is fixed throughout the entire palletizing process. No resetting is required for the hand. The reset positions of the base, the shoulder and the elbow can be detected using microswitches. By mounting three microswitches at the proper locations of the three major joints, the hard home position of the robot can be defined. These microswitches are arranged to be read from the controller. Under ontrol of the software, a motor can be stopped by a microswitch after detecting that the switch is actually closed. The status command (PRINT "I") is employed to read out the status of these three switches. A software routine is then carried out to determine whether these microswitches are open or closed. A software loop is used to move a motor one encoder hole at a time. The status command "asks" the Rhino controller each time if the microswitch under consideration has closed. If it has closed, the loop then terminates. Otherwise, the loop continues.

Because of the mechanical design of the cam and the microswitch,

the switch will close at different reset positions if the arm approaches the microswitch from different directions. To guarantee the arm will reset itself to the same hard home position no matter what direction it is moving from, a software routine has been implemented to address this problem. From experiments, it has been found that the differences of reset positions in terms of encoder holes between two opposite directions are as follows:

Switch	Joint	Difference (holes)
D	Elbow	74
E	Shoulder	91
F	Base	109

Also, it has been found that the joints will stop exactly at the desired reset positions if they are approached from the following directions.

Switch	Joint	Approaching direction	Sign
D	Elbow	Down	-
E	Shoulder	Forward	+
F	Base	Counter-clockwise	-

In case that any joint is approached from the direction opposite from the directions, the associated offset number of holes is used to carry out additional joint movement. This guarantees that the arm will reset itself to the same hard home position every time.

The following describes the reset procedure, step by step.

STEP 1: Move the arm from the current position to the hard home position at coordinate (9", 0", 12.8").

STEP 2: Read the status of the microswitches.
If any of the three switches is closed, move the associated joint away from the switch for a fixed number of encoder holes from a pre-determined direction.

STEP 3: Let DIRECTION = "-" for base motor's moving direction.

STEP 4: Let COUNT = 1.

STEP 5: Move the motor associated with the base one encoder hole. Read the status of the microswitches. If the switch related to the base motor is closed, then go to STEP 7.

STEP 6: Let COUNT = COUNT + 1.
If COUNT $\leq$ 150, go to STEP 5.
If COUNT > 150, then move the arm back to its original position. Change DIRECTION = "+", and go to STEP 4.
The maximum count of 150 corresponds to 20°, 17°, and 17° joint angles of the base, the shoulder and the elbow, respectively. From experiments, this number is sufficient to allow the Rhino robot to reach its reset position.

STEP 7: If the switch is closed with DIRECTION = "-", then go to STEP 8. Otherwise, move further the base motor for 109 encoder holes to offset the reset deviation.

STEP 8: Repeat STEPS 3 through 7 for the shoulder and the elbow with proper moving directions and offset numbers of en-

coder holes. The reset procedure is complete when all three switches D, E and F, are closed.

A BASIC program implementing the reset procedure is presented in Appendix C (subroutine starting at line 6070).

D. Palletizing Control Program

In this section, a software control program which implements the automatic palletizing task is described. This palletizing control program deals with two different conditions of box size distributions. The first assumes that the incoming box size distribution is known. The pallet pattern is changed only when a specific box size distribution changes. The multi-pallet packing approach can be implemented for the known-distribution situation.

The other condition assumes that the box size distribution is unknown. The pallet pattern may be changed whenever a pallet is full. Since the pallet pattern may be changed after every full pallet, only single-pallet packing can be employed for the unknown-distribution situation.

Compared with two-dimensional pallet packing, the required data for a three-dimensional pallet pattern is far more complicated. Elaborate consideration must be given to obtain efficient palletizing operations. The input data requirements for the palletizing control program are discussed in the section that follows.

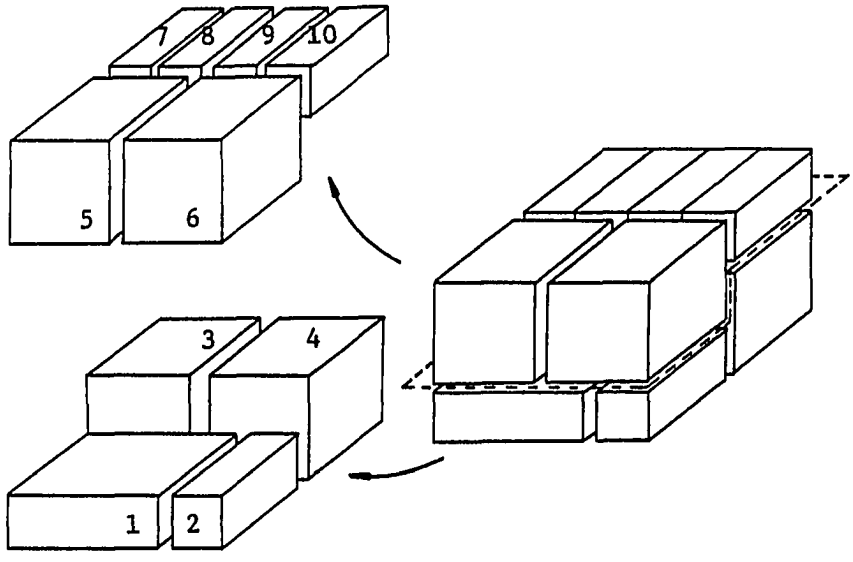
1. Data input

Unlike two-dimensional palletizing, the concept of "layer" no longer applies for the three-dimensional case. Consider a three-dimensional pallet pattern as shown in Figure 4.14. In the two-dimensional case, a layer will be placed above another layer, from the bottom to the top. However, box 5, for example, in Figure 4.14 need not wait until boxes 1, 2, 3 and 4 are placed on the pallet. Box 5 can be placed onto the pallet so long as box 1 has already been loaded. The placement sequence of box 5 is independent from any other boxes except box 1. Box 1 is therefore the only mandatory predecessor of box 5.

This is also true for box 6. So long as boxes 1 and 2 have been loaded, box 6 can be placed onto the pallet. Only boxes 1 and 2 are the predecessors of box 6.

The CPM (Critical Path Method [94]) network technique can be employed to determine the placement sequence of boxes to be loaded. The network transformation is described in the following subsection.

a. Network transformation The network consists of its nodes, represented by circles, which are used to indicate the assigned box numbers. A branch, represented by an arrow, connects two nodes and defines the predecessor and successor of two boxes. The node at the tail of the branch represents the predecessor, and the node at the head of the branch designates the successor. Three rules that must be followed in the network construction are:



Box number	Box size	Box type
1	2 x 3 x 1	3
2	1 x 2 x 1	1
3	2 x 2 x 2	2
4	2 x 2 x 2	2
5	2 x 2 x 2	2
6	2 x 2 x 2	2
7	1 x 2 x 1	1
8	1 x 2 x 1	1
9	1 x 2 x 1	1
10	1 x 2 x 1	1

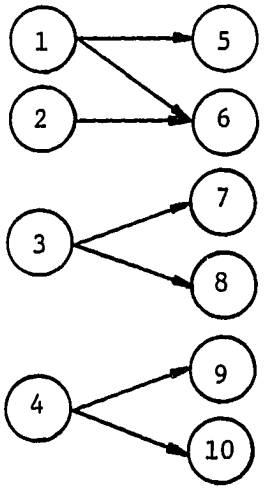
Figure 4.14. An example of a three-dimensional pallet pattern

- Each node must be identified with a unique integer number.
- The node number of a successor must be greater than its predecessor.
- A node can have many predecessors or successors. A box can be placed onto the pallet only when all its predecessors have been loaded.

The associated CPM network diagram of the previous example (see Figure 4.14) is given in Figure 4.15.

When a node has no immediate predecessor, the associated box placement location on the pallet is available for loading the specific box. The immediate predecessors of a node are updated while the palletizing proceeds. For instance, as soon as box 1 is loaded onto the pallet, node 1 is no longer the predecessor of node 6. The immediate predecessor of node 6 is reduced to node 2 only.

b. Chains of box types The placement location data for a pallet pattern are individually stored on a disk as a data file. Each record of the input data file consists of the box type and the x-, y-, and z-coordinates of the box's placement location on the pallet. It is desirable to minimize the computer search time for a desired record, and give the computer complete control of robot movements. A chain technique has been used to accomplish these tasks. A chain refers to a group of records scattered within the files and interconnected by a sequence of pointers. Boxes of the same size form a unique chain. After a box type is determined at the robot's pick-up position, the computer then searches for an adequate box placement location from the associated chain. The search procedure is continuous until a desired record is



Node	Immediate predecessor(s)
1	None
2	None
3	None
4	None
5	1
6	1,2
7	3
8	3
9	4
10	4

Figure 4.15. The network precedence diagram

found, or an end-of-chain indicator is detected. Without the chains, the computer has to search in the entire data file from the first record, and probably to the last record to determine whether or not a desired record exists.

To employ a chain, an indication is needed to show where the chain starts. The start addresses of chains can be stored in a matrix table. Consider again the previous example of Figure 4.14. The chains and pointers of the example are presented in Table 4.3.

In Table 4.3, chain 1 (box type 1) starts at record 2. The pointer of record 2 gives a number of 7; thus, the second member of chain 1 is record 7, and record 7 links to record 8 through the pointer, and so on. A pointer of number 0 indicates the end of a chain. The

Table 4.3. Chains and pointer structure[a]

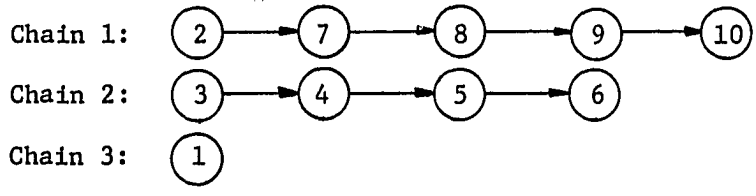
Record #	Node #	Box type	Pointer
1	1	3	0
2	2	1	7
3	3	2	4
4	4	2	5
5	5	2	6
6	6	2	0
7	7	1	8
8	8	1	9
9	9	1	10
10	10	1	0

[a]Legend: Chain 1 - - - -
Chain 2 ——
Chain 3 —·—

Chain address table:

Chain #	Box type	Start record #
1	1 (1 x 2 x 1)	2
2	2 (2 x 2 x 2)	3
3	3 (2 x 3 x 1)	1

chain members of the three chains are given in sequence below.



In this example, the initial maximum search times for a type 1 box are only 5, compared with 10 without using chains.

The pointer of each chain must be updated while the palletizing proceeds. Consider the placement of a type 1 box. If box 7 (record 7)

is placed before box 2, then the pointer of record 2 must be changed to 8, and record 7 is eliminated from chain 1. The maximum search times for a type 1 box in the subsequent process are then reduced to 4. Eventually, the length of a chain will be reduced to zero.

In case that box 2 is placed first on the pallet, the start record number of chain 1 in the chain address table (see Table 4.3) must be changed to 7. The membership of record 2 is then eliminated from chain 1. The subsequent search for a type 1 box will then start from record 7 until an available record or end-of-chain indicator is encountered. The start record number in the address table will be eventually updated to zero, which indicates no chain exists for the specific box type (i.e., no pallet space available).

For a box to be loaded, its associated number of immediate predecessors must be zero. To simplify the data input requirements, only node numbers at the tail (predecessor) and head (successor) of every network branch are stored. The number of immediate predecessors and its updating for each node is carried out by a software program. This procedure can be explained using the previous example in Figure 4.14. The node numbers of every branch's tail and head (see Figure 4.15) are listed in Table 4.4.

The node numbers in the "Tail (predecessor)" column of Table 4.4 must be in nondecreasing order. However, the order in the "Head (successor)" column can be arbitrary. A total of seven branches exist in the network. By counting the occurrence frequency of each node number

Table 4.4. The tails and heads of network branches

Record #	Tail (predecessor)	Head (successor)
1	1	5
2	1	6
3	2	6
4	3	7
5	3	8
6	4	9
7	4	10

in the "Head" column of Table 4.4., the number of immediate predecessors can be determined. These are given in Table 4.5.

Furthermore, to update the number of predecessors of each node, a predecessor address table is also generated. This address table stores each node's start record number shown in the "Tail" column of Table 4.4. Note that the first node number 3 appeared in the "Tail" column of Table 4.4 is at record number 4. Likewise, the first node number 4 is at record number 6. The predecessor address table of this example is presented in Table 4.6.

Table 4.5. Number of immediate predecessors

Node number	Number of predecessor(s)
1	0
2	0
3	0
4	0
5	1
6	2
7	1
8	1
9	1
10	1

Table 4.6. Predecessor address table

Node number	Start record number (refer to Table 4.4)
1	1
2	3
3	4
4	6
5	0
6	0
7	0
8	0
9	0
10	0

In Table 4.6, a start record address of zero indicates that its associated node number has no successors. No updating is necessary here.

Consider the following example of updating the number of predecessors. Suppose that box number 3 (node 3) has been loaded with a 2 x 2 x 2 (type 2) box. The updating procedure is as follows:

First, the predecessor address table in Table 4.6 indicates that the start record address of node 3 is at record 4. Secondly, records 4 and 5 in Table 4.4 indicate that the successors of node 3 are records 7 and 8. Finally, the number of immediate predecessors of both nodes 7 and 8 in Table 4.5 are decreased by 1, and changed to 0 and 0, respectively. Since the number of immediate predecessors of nodes 7 and 8 are altered to zero, two 1 x 2 x 1 (type 1) boxes can now be loaded.

In Appendix C, the BASIC program implementing the constructing of chains is given in statements 1960 to 2250. The updating of chain

structures corresponds to statements 3520 to 3670. The searching procedure for an available placement location on the pallet is presented in statements 3330 to 3500.

A summary of this data input procedure can be described as follows:

- Assign a unique integer number to every box placement location on a pallet.
- Transform the placement relationships of boxes to a CPM network precedence diagram.
- Input the following data:

  1) number of box types, number of nodes, number of branches

  2) node #, box type, x-coordinate, y-coordinate, z-coordinate, orientation

     The total number of records in this category should equal the number of nodes specified in 1)

     Each record consists of the following attributes.

     node # : node number assigned to network diagram

     box type : associated box type of the node number

     x-coordinate : x coordinate value of a box location on a pallet

     y-coordinate : y coordinate value of a box location on a pallet

     z-coordinate : z coordinate value of a box location on a pallet

     orientation : placement orientation of a box;
     do nothing if 0;
     roll the hand 90° if 1.

     It is assumed that boxes conveyed to the robotic palletizing cell follow a fixed orientation.

  3) tail of branch, head of a branch

     The total number of records in this category should equal the number of branches specified in 1).

2. Palletizing procedure

A miniature robotic palletizing station has been constructed using the Rhino XR-2 robot, a conveyor, and a turntable. The system layout of this robotic palletizing cell is illustrated in Figure 4.16.

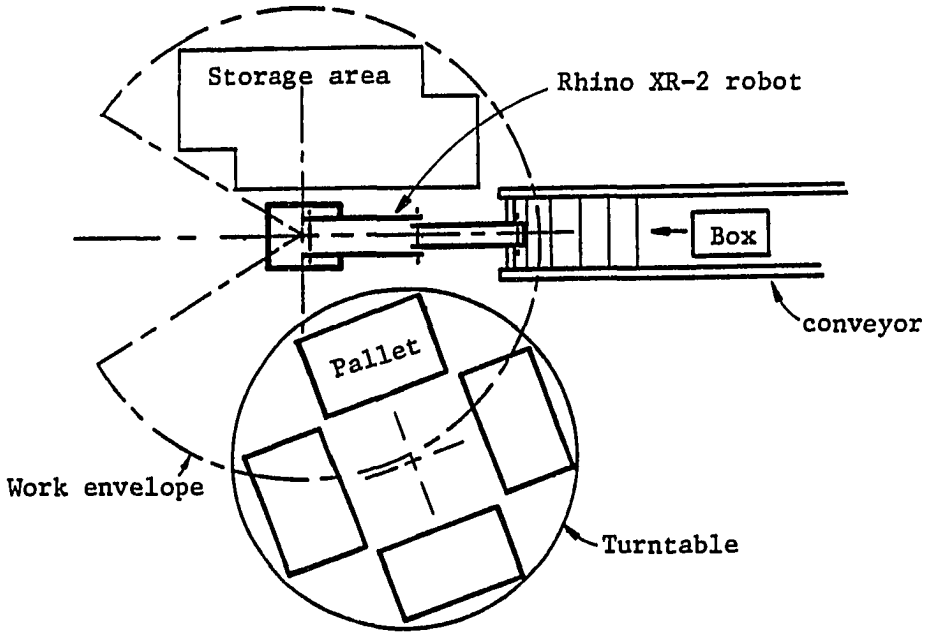


Figure 4.16. System layout of the robotic palletizing cell

The palletizing process proceeds, step by step, as follows:

1. Move the robot arm until the gripper is right above the box at the pick-up position.

2. Lower the robot and actuate the gripper to pick up the box.

3. Raise the arm clear of the in-feeding conveyor, so that it is above the height of any obstruction.

4. If a pallet space is available for the box, move the arm to the proper location above the pallet. Otherwise, move the box to the proper location above the box's associated storage area.

5. Lower the robot arm straight down so that it will not contact other loaded boxes. Release the gripper and place the box.

6. Raise the arm to a height above any obstruction.

7. For every off-line storage area, if the storage area is not empty and a pallet space is available for this specific box size, then remove one and only one box from the storage area and place it onto the pallet. This step is repeated until all storage areas are checked.

8. Go to 1.

Steps 1 through 8 are also defined as a palletizing cycle from pick-up to pick-up. The complete palletizing procedure is illustrated with a block flow diagram as shown in Figure 4.17. The following descriptions explain the operations of selected blocks. The block numbers refer to the numbers assigned to the blocks in Figure 4.17. Circles A and B presented in Figure 4.17 will be explained later.

BLOCK 1: Initially, the arm is set to its hard home position through the manipulation of keyboard.

BLOCK 2: The placement location data of a pallet pattern is input to the control program. The chain for every box type is also constructed as described in the previous section.

Each pallet pattern is stored as a separate data file in the disk. The data are read into the computer memory only when they are needed. Therefore, the number of pallet patterns that can be stored are constrained only by available space on the floppy disk.

BLOCK 3: From experiments, the Rhino XR-2 robot must self-reset to the hard home position for every five palletizing cycles from pick-up to pick-up.

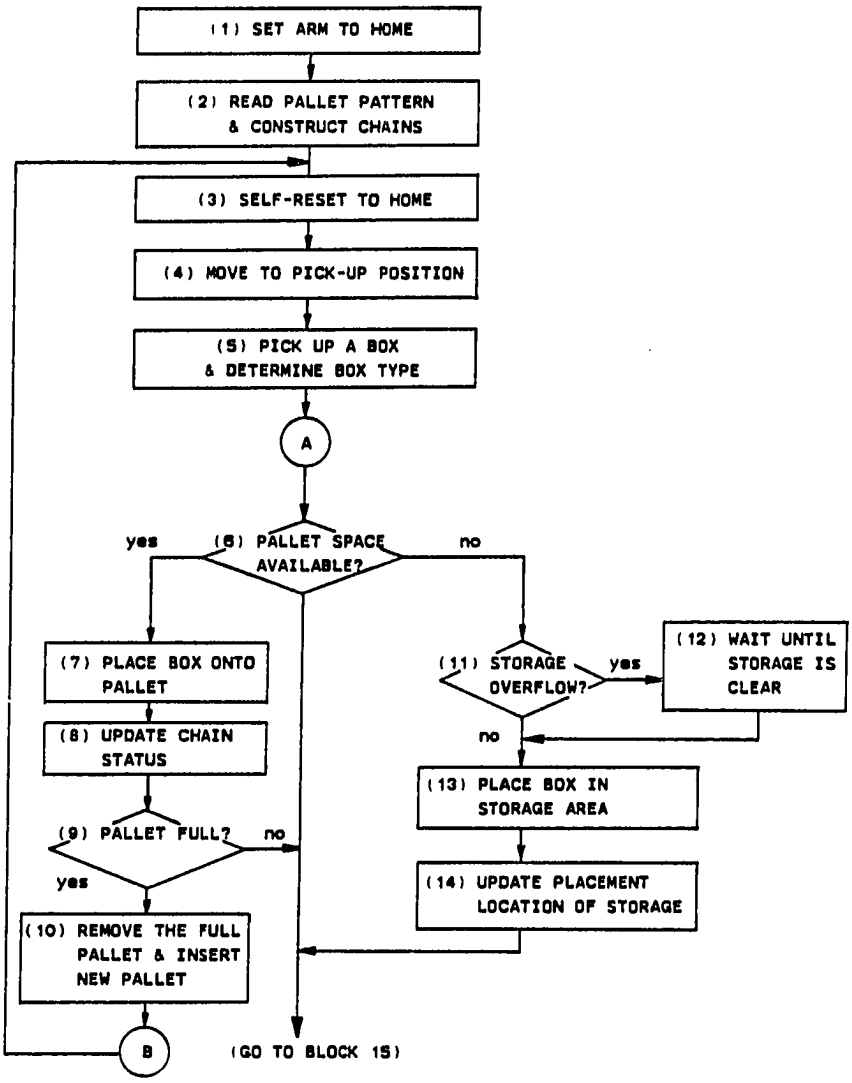


Figure 4.17. Block flow diagram of palletizing procedure

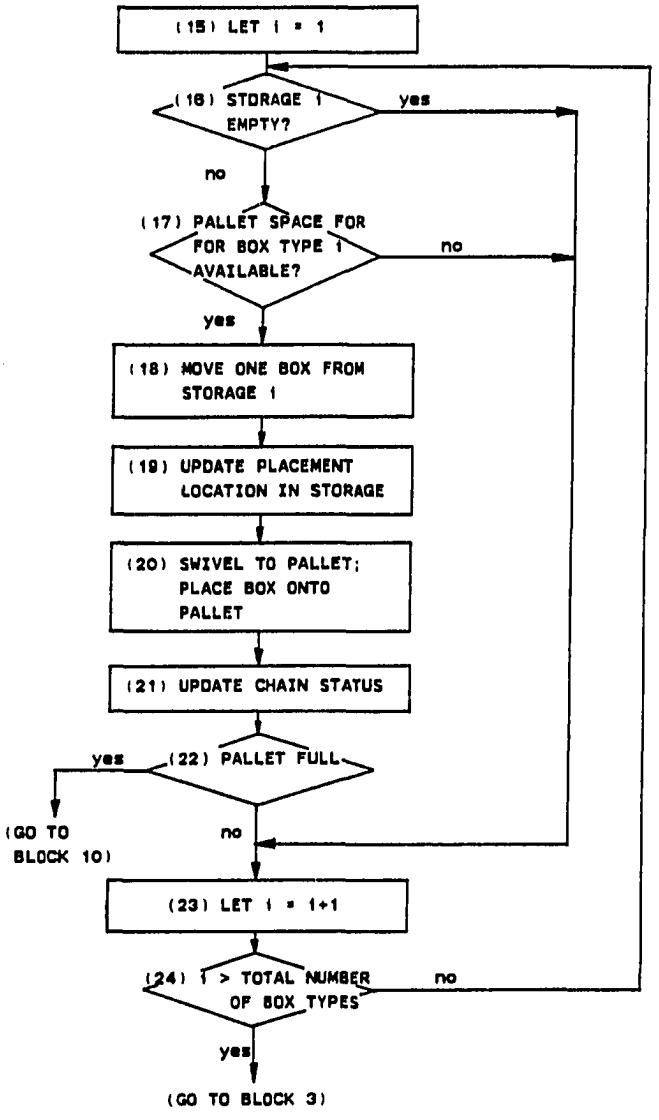


Figure 4.17. continued

BLOCK 5: In industrial applications, the automatic identification systems such as bar code readers and machine vision can be employed to identify the box sizes and signal the robot for proper operations. In this research, a random number generator is used to generate box types. No actual measurement of box sizes is carried out. However, the random number generator has been adequate to serve the purpose of simulating an automatic identification device. The algorithm used to generate random box types will be described in detail in Chapter V.

BLOCKS 6, 7 and 8: The computer searches for a pallet space from the box's associated chain. If any member of the chain has zero number of predecessors, a pallet space is available for the box just picked up. The chain is then updated as described previously.

BLOCKS 9 and 10: When a pallet is full, a fork-lift truck or a automatic transfer removes the full pallet and inserts a new empty pallet. The chain status must be restored if the same pallet pattern is to be used for the next pallet. For multi-pallet packing with a turntable, a lowest priority is assigned to the newly inserted pallet. The priority numbers are increased for those remaining pallets which are not completely loaded. Only one pallet pattern is required for the palletizing control program no matter how many simultaneously loaded pallets are on the turntable. Every pallet has exactly the same pallet pattern. It is retrievable by properly rotating the turntable.

BLOCKS 11 and 12: Off-line storage areas are used for those boxes that cannot be immediately loaded as they arrive at the robot's pick-up

position. In case that the storage area overflows, the robot will stop its operations. The palletizing process will resume after the excess number of boxes in the storage area are manually removed to an acceptable level.

BLOCK 14: Each storage area is of cubic shape, and stores only identical boxes. Only the x, y, z coordinates of the bottom rightmost corner (initial placement location) and the top leftmost corner (extreme placement location) of the storage area need to be known by the robot. Every new placement location can be obtained either by increasing the width or the length of the box from the coordinate of the previous placement location. When one extreme point along the x-axis is reached, the value of the y coordinate is increased by the box's width (or length, depending on the box's placement orientation). The x coordinate value is reset to its initial value.

Once both x and y coordinates reach their extreme points, the value of the z coordinate is increased by the box's height. The value of the x and y coordinates are then reset to their initial values. This enables the storage process to continue with minimal computer memory requirements.

When the x, y and z coordinates reach simultaneously their extreme points, an indication of storage overflow will be notified by beeping and flashing the information on the display.

BLOCKS 15 through 24: After a box is picked up from the in-feeding conveyor and placed either on the pallet or in the storage area,

the robot will remove one box from every storage area and place it onto the pallet so long as the storage area is not empty and a pallet space is available. Boxes are removed from storage areas according to LIFO (Last-in First-out) discipline.

A BASIC program implementing the palletizing procedure is presented in Appendix C. This palletizing control program is further developed to manage two conditions of box size distributions. One is a known box size distribution. The other is an unknown box size distribution. They are discussed in the subsections that follow.

a. Known distributions With a known distribution, it is assumed that the box size proportions and the length (total number of boxes) of a distribution run are known. The sequence of box sizes arriving at the robot cell is still random, but the total number of boxes of each size can be pre-determined. The number of boxes may be determined from customer orders or master production schedules. Whenever the end of a box size distribution is detected by counting the total number of boxes loaded, the computer will switch the pallet pattern for a new distribution run. Circle A in Figure 4.17 carries out this task. When an end-of-distribution is detected, a new pallet pattern number will be determined and circle A routes back to BLOCK 2. A multi-pallet packing approach can be applied for the situation of known size distributions.

b. Unknown distributions With an unknown distribution, it is assumed that the information of a box size distribution cannot be ob-

tained before palletizing starts. The total number of boxes of a distribution run cannot be pre-determined. Whenever a pallet is full, the computer will select the best "match" pallet pattern according to the boxes in the look-ahead queue (in-feeding conveyor). The multi-pallet packing cannot be used for the situation of unknown size distributions. Consider the following situation of applying the double-pallet packing for unknown box size distributions.

After the pallet of the higher priority is full, an empty pallet is inserted, and a new pallet pattern is selected for this empty pallet according to the boxes on the in-feeding conveyor. Meanwhile, the pallet of the lower priority may have been loaded with some boxes. The new pallet pattern may not fit this partially loaded pallet. This pallet must use the original pallet pattern and wait for boxes of appropriate sizes to fill up the remaining space. It is possible that the remaining spaces may remain empty for a long period of time. This is because the box size distribution may have been changed since the pallet of the higher priority is full. Therefore, only single-pallet packing can be employed for the situation of unknown distributions.

Both conditions of known and unknown distributions have been simulated. The simulation results will be discussed in Chapter V.

3. "Match" selection

For the situation of unknown size distributions, the procedure of selecting a best "match" pallet pattern according to the boxes in the look-ahead queue is described, step by step, as follows:

STEP 1: Use eq. (4.1) to determine the look-ahead queue-length.

The equation is rewritten below.

$$\sum_{i=1}^{Q-1} v_i < \alpha V \leq \sum_{i=1}^{Q} v_i$$

where v_i = volume of i^{th} box in the queue;

V = volume of a pallet;

Q = last box in the queue;

= total number of boxes in the observed queue.

STEP 2: For boxes $v_1, v_2, \ldots, v_Q$, compute the frequency of each box size. Denote the frequency of box size i as FREQ(i).

STEP 3: Calculate the cumulative deviation of box ratios. Assume a total of N pallet patterns are stored on the disk.

For j = 1, 2, ..., N, compute

$$DEV(j) = \sum_{i} \left| FREQ(i)/Q - NBOX(i,j)/TOTAL(j) \right| \qquad (4.9)$$

where $|\cdot|$ gives the absolute value

DEV(j) = cumulative deviations of box ratios for pallet pattern j

NBOX(i,j) = the number of boxes of size i in pallet pattern j

$TOTAL(j) = \sum_{i} NBOX(i,j)$

= total number of boxes in pallet pattern j

FREQ(i)/Q = box ratio of size i in the look-ahead queue

NBOX(i,j)/TOTAL(j) = box ratio of size i in pallet pattern j

STEP 4: The N stored pallet patterns are divided into three groups according to their associated cumulative deviations of box ratios. The first group consists of the pallet patterns that

FREQ(i) = 0 and NBOX(i,j) > 0, for any i.

The second group consists of the pallet patterns that

FREQ(i) < NBOX(i,j), or

FREQ(i) > 0 and NBOX(i,j) = 0, for any i.

The remaining pallet patterns are in the third group. For every pallet pattern j in group 3,

FREQ(i) $\geq$ NBOX(i,j), for all box size i.

This means that the total number of boxes in the look-ahead queue is sufficient to complete a full pallet load. Pallet patterns in group 2 cannot fill a full pallet load in this observed look-ahead queue. They have to wait for appropriate box sizes after all boxes in this look-ahead queue are packed. When FREQ(i) = 0, the pallet patterns which have a nonzero number of size i boxes are separated from group 2. This prevents the computer selecting a pallet pattern that requests a specific box size which may not arrive at the robotic palletizing cell.

Therefore, the pallet patterns in group 3 have the highest priority; the ones in group 2 have the second priority; and the ones in group 1 have the lowest priority.

STEP 5: Select the pallet pattern whose cumulative deviation of box ratios is minimum in group 3. If no such pallet pattern exists in group 3, select the pallet pattern that has minimum cumulative deviation of box ratios in group 2. If no pallet patterns satisfy the requirements in both groups 2 and 3, the pallet pattern has minimum cumulative deviation of box ratios is selected from group 3.

The evaluation of this procedure's performance is carried out using simulation in Chapter V.

This procedure is applied in the robotic palletizing program, shown as circle B in Figure 4.17 for the situation of unknown size distributions. After circle B, the flow routes back to BLOCK 2 in Figure 4.17 to read into placement location data of the new selected pallet pattern. A BASIC program implementing the dynamic determination of a best "match" pallet pattern is given in Appendix E (subroutine starting at line 7340). In addition, two complete robotic palletizing programs, one for known distributions and the other for unknown distributions, are presented in Appendixes D and E, respectively.

To evaluate the feasibility and performance of the robotic palletizing system, two simulations have been completed. One is for multi-pallet packing with up to four simultaneously loaded pallets. The other is for determining the best value of the look-ahead factor (α in eq. 4.1). Palletizing efficiencies of known and unknown distributions are also compared based on the two simulation results. They are the subjects of the following chapter.

V. THE PALLETIZING SIMULATION

A. Introduction

The developed simulation model can be used for design, procedural analysis and performance. In this research, two major simulation results are presented. One is the simulation statistics of multi-pallet packing with a turntable. Single-, double-, triple-, and quadruple-pallet packing have been evaluated. This refers to the simultaneous loading of one, two, three, and four pallets, respectively. The second set of simulation statistics are for palletizing with unknown box size distributions. Five different look-ahead factors, from 1 to 3 in increments of 0.5, have been examined in this simulation. With single pallet packing, simulation results of both known and unknown box size distributions are also collected. A "known" box size distribution means that the length of each distribution run and box proportion of each type is known by the palletizing software program. In contrast, an "unknown" distribution means that all information cannot be pre-determined (refer to Chapter IV). This allows the performance of the dynamic selection procedure for the best "match" pallet pattern to be evaluated.

The miniature physical simulator of the robotic palletizing cell as described in Chapter IV is employed to collect all required palletizing data. The physical simulator has served the following four purposes.

- Demonstration of the robotic palletizing operations.
- Verification of the validity of the simulation programs.
- Collection of palletizing statistics.

- Evaluation of the feasibility and performance of the robotic palletizing system.

B. Condition Setups

This section describes the palletizing conditions for the simulation. These consist of box sizes, pallet size, the arrival pattern of boxes, box size distributions, length of a distribution run, box size sequence in a distribution, and sequence of distributions.

1. Assumptions

To simplify the simulation process, the following assumptions have been applied.

- Boxes placed on the in-feeding conveyor follow a fixed orientation. The longest dimension of a box's length and width must be in the direction oriented along the conveyor's length axis. This may be also required in industrial applications since the robot should not need to make a series of orientation moves when picking up a box. To minimize cycle time, boxes should be presented to the robot with consistent orientation and positioning [89]. This also helps simplify the system.

- A random number generator, which randomly generates box sizes, is used to simulate the operation of a bar code reader. In industrial applications, the bar code reader or scanning device can be employed to identify box sizes and signal the robot for proper operations.

- The capacity of off-line storage areas is infinite. This permits the collection of queue statistics without constraints of physical storage space in the miniature model. The simulation process will not be interrupted because of storage overflow. Also, the maximum number of boxes ever stored in the storage area at a time can be also determined with this space relaxation.

The simulation conducted in this research consists of the following four box sizes:

Size 1: 1" x 1" x 1",

Size 2: 1" x 2" x 1",

Size 3: 2" x 2" x 2",

Size 4: 2" x 3" x 1".

A 4" x 4" pallet is employed to load these boxes with a stacking height limit of 4 inches.

It is assumed that boxes arrive at a rate such that they are always available at the pick-up position when the robot is ready to pick up a box.

2. Box size distributions

Boxes arrive at the robot's pick-up position according to 20 various distributions of the combinations of the above four box sizes. The 20 box size distributions (proportions) are listed in Table 5.1.

The four box sizes are divided into thirds. The distributions range from 100% of one box size, 33.3% and 66.7% of two box sizes, to 33.3% of three box sizes. The 20 distributions in Table 5.1 represent all possible distribution combinations by dividing the four box sizes into thirds. These 20 distributions have been used by Fleming [33] for simulating the palletizing process with two-dimensional pallet patterns.

The heuristic dynamic programming algorithm described previously in Chapter III is employed to solve for the pallet pattern for each of the 20 distributions. In this simulation, it has been determined that the box size proportions of a pallet pattern should be as close as possible to those shown in Table 5.1. The 20 pallet patterns are de-

Table 5.1. 20 box size distributions

Distribution number	Proportion of boxes			
	Size 1 (1 x 1 x 1)	Size 2 (1 x 2 x 1)	Size 3 (2 x 2 x 2)	Size 4 (2 x 3 x 1)
1	0	1/3	1/3	1/3
2	0	1/3	2/3	0
3	1/3	0	1/3	1/3
4	0	2/3	1/3	0
5	1/3	1/3	1/3	0
6	0	2/3	0	1/3
7	1/3	0	2/3	0
8	1/3	1/3	0	1/3
9	0	1/3	0	2/3
10	0	0	1/3	2/3
11	1/3	2/3	0	0
12	0	0	0	1
13	1/3	0	0	2/3
14	0	0	1	0
15	0	1	0	0
16	0	0	2/3	1/3
17	2/3	1/3	0	0
18	1	0	0	0
19	2/3	0	1/3	0
20	2/3	0	0	1/3

termined based on this criterion. Some pallet space must be sacrificed to obtain desired box size proportions. The heuristic dynamic programming procedure that determines the required number of boxes of each size of each distribution is presented in Appendix F. The associated pallet patterns and their precedence diagrams are presented in Appendix G.

The determined numbers of boxes for each distribution are summarized in Table 5.2. The numbers shown in this table give the required number of boxes of each type for each pallet pattern. Each box size distribution has its own associated pallet pattern. For instance, distribution 1 uses pallet pattern 1 which consists of four 1 x 2 x 1 boxes, four 2 x 2 x 2 boxes, and four 2 x 3 x 1 boxes. For a full pallet load, pattern 1 will have 12 boxes.

3. Length of a distribution run

Decision analysis based on the results of a simulation model normally requires an estimate of the average simulation response. Steady-state behavior of a system specifies that the probability mechanism describing the variability is unchanging and is no longer affected by the starting condition. In this simulation, steady-state means that the quantities of each box size in the associated off-line storage area have achieved relatively stable levels. The length of a distribution run should be long enough so that the steady-state behavior can be observed. However, based on the preliminary simulation run using pallet patterns (distributions) 1 and 3, no steady-state has been observed even when

Table 5.2. Number of boxes used for 20 pallet patterns

Distribution/ pallet pattern number	Number of boxes				Total # of boxes in pallet cube
	Size 1 (1 x 1 x 1)	Size 2 (1 x 2 x 1)	Size 3 (2 x 2 x 2)	Size 4 (2 x 3 x 1)	
1	0	4	4	4	12
2	0	4	7	0	11
3	4	0	4	4	12
4	0	10	5	0	15
5	5	5	5	0	15
6	0	12	0	6	18
7	4	0	7	0	11
8	7	7	0	7	21
9	0	4	0	8	12
10	0	0	2	4	6
11	12	25	0	0	37
12	0	0	0	8	8
13	4	0	0	8	12
14	0	0	8	0	8
15	0	32	0	0	32
16	0	0	4	2	6
17	32	16	0	0	48
18	64	0	0	0	64
19	12	0	6	0	18
20	16	0	0	8	24

the length of the distribution run is increased up to 1,200 boxes. Figure 5.1 illustrates the variation of 2" x 3" x 1" boxes (from distribution 1) in the storage area. Notice that although no steady-state is found in Figure 5.1, there is a trend forming a cycle for every 200 boxes placed. At the beginning of a cycle, the stored boxes in the storage area is about at the zero level. Then the quantities of boxes in the storage area increase and vary within the cycle. Finally, at the end of the cycle, the quantities of stored boxes again reach the zero level. This trend is also observed for box sizes 1 x 1 x 1, 1 x 2 x 1 and 2 x 2 x 2. The variations of stored boxes for these three box sizes are presented in Appendix H.

In industrial applications, a loading sequence of 1,200 boxes is unrealistically long. Therefore, it was decided that each distribution run should consist of 200 boxes. There are 20 different distributions. A total of 4,000 boxes are therefore "loaded" in each simulation run.

Since no steady-state level of stored boxes in storage areas is observed, the simulation starts with an empty system. That is, the four off-line storage areas are initially empty when a simulation run begins.

4. Box size sequence in a distribution

One of the purposes of the conducted simulation is to compare alternative system configurations, such as single- vs. multi-pallet packing, to find which method best satisfies a given objective. The way a system is configured can be thought of as an independent variable. If the palletizing alternatives are to be correctly evaluated, all other

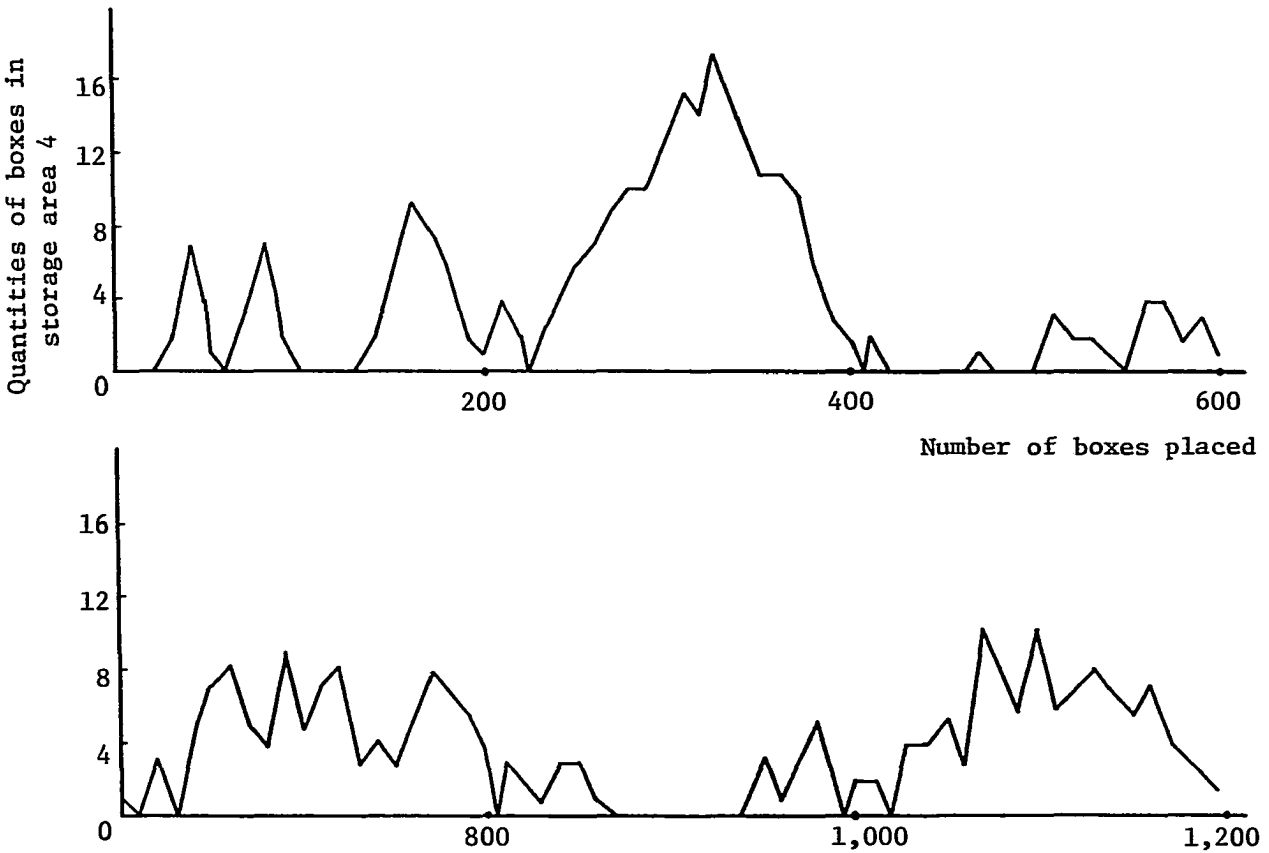


Figure 5.1. Variation of boxes stored in the storage area - box size 2 x 3 x 1, distribution #1

conditions under which the alternatives are investigated should be identical from experiment to experiment. The randomness of box size sequence in a distribution has a potential influence on the measure of palletizing behavior. Under this consideration, the box size sequence in each of 20 distributions are predetermined. The same sequence of box sizes are carried for all experiments in this simulation.

A random number generator was used to generate the sequence of box sizes for each of the 20 distributions. To assure the box size proportions generated by the random number generator are exactly the same as those specified in Table 5.1, the following nonsequential distribution sampling algorithm was applied [67].

Assume that box size sequence of distribution 1 is to be generated. The box size proportion of distribution 1 and the corresponding numbers of boxes out of a total of 200 boxes are:

Box size	Size number	Proportion	Number of boxes	Cumulative boxes
1 x 2 x 1	2	1/3	66	66
2 x 2 x 2	3	1/3	67	133
2 x 3 x 1	4	1/3	67	200

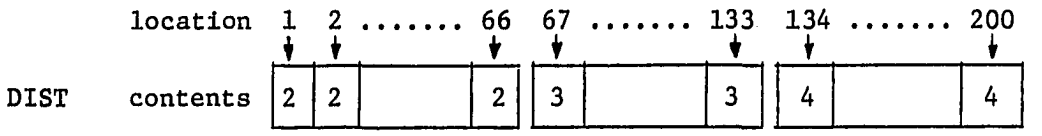
Therefore, the random number generator must generate exactly 66, 67 and 67 boxes for size numbers 2, 3 and 4, respectively.

Consider an array named DIST of length 200 which contains the integers 2, 3 and 4 according to the following locations.

Contents of DIST at locations 1 through 66 = 2.

Contents of DIST at locations 67 through 133 = 3.

Contents of DIST at location 134 through 200 = 4.



Then, perform the following procedure.

STEP 0: Let N = 200.

STEP 1: Choose a random integer J in the range [1, N] (i.e., from 1 to N, inclusively),

$$J = INT[RAN * N] + 1,$$

where function INT returns a largest integer number no greater than the value in the brackets. RAN is any random number generator that gives a random value between 0 and 1, exclusively.

STEP 2: Choose the contents of DIST(J) as the Nth sample value of the distribution. Replace the contents of DIST(J) by the contents of DIST(N).

STEP 3: Let N = N - 1. If N = 0, then terminate the procedure. Otherwise, go to STEP 1.

The above process generates the desired number of boxes for each size, and randomly distributed sequence of box sizes.

5. Permutation of 20 distributions

It is desired to make the incoming box distribution transitions to the physical simulator change radically so that the response and feasibility of the robotic palletizing system can be examined.

The most severe demands on off-line storage space may occur at the transition point between the change of two different distributions. In this simulation, the worst-case permutation of the 20 distributions is applied in terms of the cumulative variations of box size proportions. The cumulative variations of box size proportions are defined as the cumulative differences of box size proportions between two adjacent distributions. For a worst-case permutation of distributions, the summation of the cumulative variations of box size proportions is the maximum among all possible orders of distributions.

Table 5.3 presents one of the worst-case permutations of the 20 box distributions. The maximum difference of box size proportions between any two distributions is two (2). For distributions 1, 3, 5 and 8 (three box sizes, 1/3 proportion for each), the maximum difference of box size proportions between any of these four distributions and any other distributions is 1 1/3. Therefore, the sequence of 20 box size distributions in Table 5.3 gives the maximum cumulative variations of box size proportions. This is the order used in this simulation.

6. Summary of condition setups

All palletizing conditions used in the simulation are the same for both known and unknown box size distributions. The definition and palletizing procedures for known and unknown distributions have been described in detail in Chapter IV. The setups are summarized as follows:

- Four box sizes used: 1" x 1" x 1" (size 1), 1" x 2" x 1" (size 2), 2" x 2" x 2" (size 3) and 2" x 3" x 1" (size 4).

Table 5.3. Worst-case permutation of box distributions

Distribution number	Box size				Cumulative variations
	1 1 x 1 x 1	2 1 x 2 x 1	3 2 x 2 x 2	4 2 x 3 x 1	
1	0	1/3	1/3	1/3	-
18	1	0	0	0	2
4	0	2/3	1/3	0	2
13	1/3	0	0	2/3	2
2	0	1/3	2/3	0	2
3	1/3	0	1/3	1/3	1 1/3
15	0	1	0	0	2
19	2/3	0	1/3	0	2
9	0	1/3	0	2/3	2
7	1/3	0	2/3	0	2
6	0	2/3	0	1/3	2
5	1/3	1/3	1/3	0	1 1/3
12	0	0	0	1	2
17	2/3	1/3	0	0	2
16	0	0	2/3	1/3	2
11	1/3	2/3	0	0	2
10	0	0	1/3	2/3	2
8	1/3	1/3	0	1/3	1 1/3
14	0	0	1	0	2
20	2/3	0	0	1/3	2
				Total	36

- Pallet dimensions: 4" x 4" with a stacking height limit of 4 inches.
- Box arrival rate: always available on the in-feeding conveyor when the robot is ready to pick up a box.
- Twenty different box size distributions (see Table 5.1).
- Twenty associated pre-determined pallet patterns (see Appendix G).
- Length of a distribution run: 200 boxes.
- A total of 4,000 boxes for each simulation run.
- Sequence of 20 box distributions: worst-case permutation (see Table 5.3).
- Simulation starts with an empty system.
- Simulation is terminated as soon as the placement of 4,000 boxes is complete.
- The layout: the system layout of the robotic palletizing cell used in this simulation is illustrated in Figure 5.2. This gives the detailed locations of four storage areas, the Rhino robot, the conveyor and the 4-pallet turntable.

7. Collection of robot movement times

From a preliminary simulation study, it was estimated that each simulation run of 4,000 boxes required about twenty-five hours. There are nine experiments[1] conducted in this simulation. To simplify the process of data collection, the movement times of the Rhino robot are collected. When simulation starts, the Rhino robot is shut off and a built-in timer of the T1 Professional microcomputer is employed to

[1]These nine experiments are single-, double-, triple- and quadruple-pallet packing for known distributions, and look-ahead factors of 1.0, 1.5, 2.0, 2.5 and 3.0 for unknown box size distributions.

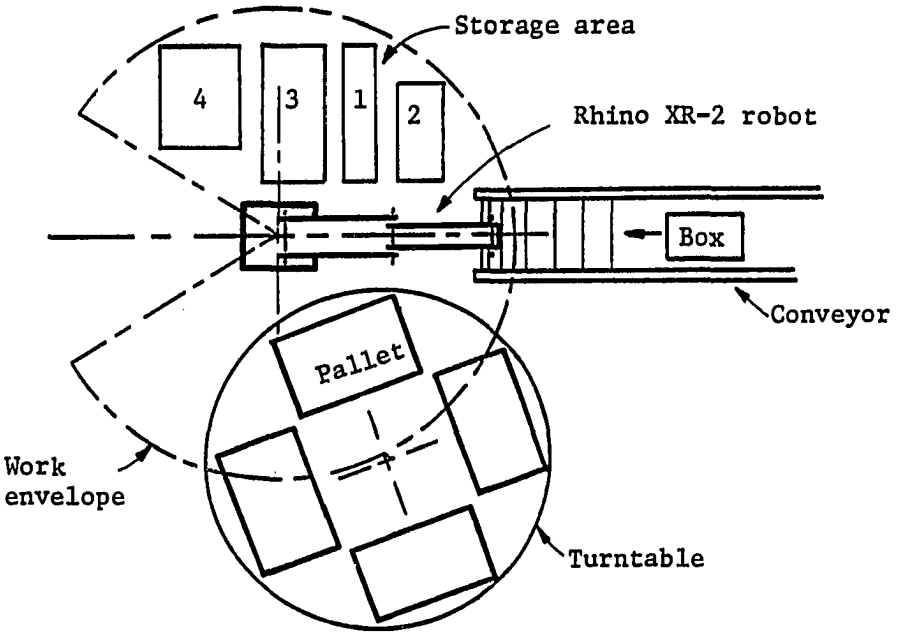


Figure 5.2. System layout of the physical simulator

simulate the actual robot movement times. For example, assume that the robot movement from the pick-up position to the storage area requires 5 seconds. When the palletizing software program proceeds to this operation, the program will stop for 5 seconds. The process will resume after 5 second delay is reached. In this way, no human attendance was necessary for the entire palletizing simulation.

The physical simulator of the robotic palletizing cell was used to collect all required movement times of the Rhino robot and the turntable. At least two full pallets for each distribution were manually

observed to verify the validity of the pallet patterns and the logic flow of the palletizing process. Robot movement times were also collected during this observation using the built-in timer of the Tl microcomputer. To obtain accurate robot movement times, the palletizing operation was divided into eleven detailed motions. All movement times of the eleven motions are deterministic rather than probabilistic. These collected movement times were substituted for the actual robot movements in the simulation programs. Appendix I presents the eleven detailed motions and collected movement times.

C. Evaluation Criteria

The criteria used to evaluate the performance and feasibility of the robotic palletizing system are based on the queue statistics of stored boxes in the off-line storage areas, total palletizing time of 4,000 boxes and robot operation times of moving boxes to and from the storage areas. All these statistics are collected for each of the 20 distributions and the total simulation of 4,000 boxes. The built-in timer of the Tl microcomputer is used to collect all time dependent statistics while the simulation proceeds. This section describes how the statistical measures are defined and how they are derived.

1. Loading statistics

The loading criteria measure the total number of boxes loaded onto the pallet and the various palletizing times. They consist of the following.

TOTAL BOXES LOADED: total number of boxes loaded onto the pallets. A counter is used to record this number. It is increased by one whenever a box is placed onto the pallet either from the in-feeding conveyor or the storage area.

TOTAL PALLETIZING TIME: total time required to complete the placement of 4,000 boxes (or 200 boxes of a distribution run) either on pallets or storage areas. This is obtained by subtracting the start time from the terminating time of a simulation run.

AVERAGE CYCLE TIME: the average time required from pick-up to pick-up. A palletizing cycle is as follows: Pick up a box from the in-feeding conveyor, and place it onto a pallet, if possible. Otherwise, place it in the off-line storage area. Then try to remove one box from each storage area and place it onto the pallet. Finally, move the arm back to the pick-up position.

The complete operation sequence of a palletizing cycle has been defined in section D.2 of Chapter IV. The average cycle time is obtained by dividing the total cumulative cycle times by the total number of cycles.

TOTAL OPERATION TIME IN STORAGE: total process time spent moving boxes to and from the storage areas. The larger the operation time in the storage areas, the less efficient the palletizing process. An ideal palletizing operation will have zero operation time in the storage areas. This ideal palletizing time can be used as the standard to evaluate the performance of other palletizing alternatives.

An ideal cycle time is the following. The robot picks up a box from the pick-up position on the conveyor and directly places it onto a pallet. Then the robot moves back to the pick-up position. The total operation time in storage areas is computed by subtracting the cumulative ideal cycle times from the total palletizing time.

2. Queues in storage areas

The operating characteristics of the queues used [85,101] are applied to evaluate the performance of the robotic palletizing system. The operating characteristics employed consist of the following.

TOTAL BOXES GENERATED: total quantities of boxes of each size that have been present on the conveyor and been picked up by the robot. A counter for each box size is used to record this number. Whenever a box size is generated by the random number generator, the corresponding counter is increased by one.

TOTAL ENTRIES: total number of boxes which enter a storage area over the duration of the simulation period. TOTAL ENTRIES is the value of a counter initialized by zero, and incremented by one whenever a box is placed into the storage area.

ZERO ENTRIES: total number of boxes which can be immediately placed onto pallets after being picked up from the conveyor. These boxes spend zero residence time in the storage area. ZERO ENTRIES can be obtained by subtracting TOTAL ENTRIES from TOTAL BOXES GENERATED.

MAXIMUM CONTENTS: largest quantities of boxes ever stored in a storage area at a time. This value can determine whether a storage area

is sufficient in size. It is desired to have the value of MAXIMUM CONTENTS as small as possible so that the storage overflow will not occur during palletizing. A variable initialized by zero is used to record the number. Whenever the current number of boxes in the queue is larger than the one previously recorded, the value of the variable is updated.

CURRENT CONTENTS: total number of boxes left in the storage area when the simulation is terminated. A counter is used to update this value. When a box is placed into the storage area, the counter is increased by one. When a box is removed from the storage area, the counter is decreased by one.

AVERAGE CONTENTS: average number of boxes residing in the storage area at any time. This value is determined as follows:

$$\text{Average contents} = \Sigma\, N \cdot P(N),$$

where N = queue length

= number of boxes stored in the storage area

= 0, 1, 2, ...

P(N) = the probability that there are N boxes in the storage area

$$= \frac{\text{total waiting times with queue length N}}{\text{total palletizing time}}$$

AVERAGE WAITING TIME: average time that a box spends in the storage area. This value is determined as follows:

$$\text{Average waiting time} = \frac{\Sigma \text{ waiting time of a box in the queue}}{\text{total boxes generated}}$$

Notice that the above expression includes the zero entries, the

boxes that can be immediately loaded onto pallets.

The standard deviations of the number of boxes in the storage area over time, and the waiting time per box do not have contributed meaning for this research. They are not collected in the simulation.

D. Simulation Results

In this section, three major simulation results are discussed. The first is the simulation multi-pallet packing with known box size distributions. The second is the simulation of the dynamic selection for a best "match" pallet pattern with unknown box size distributions. The effect of alternate look-ahead factors is examined. The third set of simulation results compares the palletizing performance between known and unknown box distributions.

Before discussing these three simulation results, the previously described worst-case permutation of the 20 distributions is evaluated.

1. Worst-case permutation of distributions

In Table 5.3, the worst-case permutation of 20 distributions is determined according to the maximum cumulative variations of box proportions. To ensure this distribution permutation places severe demands on storage space at the transition point between different distributions, the palletizing statistics of the worst-case permutation are compared with those of 30 random sequences of distributions. The 30 sequences of randomly generated distribution numbers are presented in Appendix J. Since each simulation run may require twenty-five hours, the robot move-

ment times are set to zero, and only time independent statistics are collected. The statistics to be compared are:

- Maximum contents: This gives the largest quantity of boxes ever stored in the storage area at a time. It can be used to determine the required storage space. Since the robot's work envelope is limited, a smaller value of maximum contents is desirable. Also, the most severe demands on box storage space may occur at the transition points between two adjacent distributions. The maximum contents statistic can be employed to observe whether there is any significant change of the demands on storage space.

- Zero entries: This gives the total number of boxes which can be directly loaded onto pallets. A larger amount of zero entries indicates that less total operation times in storage areas may be required since the movements of moving the robot arm to and from storage areas are reduced.

This simulation has used single-pallet packing with known box size distributions. The simulation results of the 30 random distribution sequences along with the worst-case permutations are presented in Tables 5.4a and b. Table 5.4a shows the statistics of maximum contents, and Table 5.4b presents the statistics of zero entries.

Scanning Table 5.4a, notice that the maximum contents in storage area 1 range from 12 to 15, which gives an average of 13.5, compared with 12 of the worst-case permutation. The maximum contents in storage area 2 consistently fall between 9 and 10, which gives an average of 9.77. This compares with 10 of the worst-case permutation. The maximum contents in storage area 4 are consistently between 25 and 26, which gives an average of 25.2. This compares with 25 of the worst-case permutation. The differences of maximum contents between the worst-case permutation and 30 random runs for storage areas 1, 2 and 4 are not significant. The

Table 5.4a. Comparison of maximum contents

Run	Maximum contents, Storage area 1	2	3	4
Worst-case	12	10	44	25
1	12	10	27	25
2	15	10	31	25
3	12	10	22	25
4	15	9	31	25
5	15	10	37	26
6	12	9	30	25
7	13	10	27	25
8	13	10	25	26
9	15	10	32	26
10	13	10	31	25
11	12	10	42	25
12	12	10	24	25
13	13	10	37	26
14	15	10	33	25
15	12	10	32	25
16	13	10	30	25
17	15	10	32	25
18	15	9	38	25
19	12	9	23	26
20	15	10	39	25
21	13	9	37	26
22	15	10	43	25
23	15	10	37	25
24	12	10	45	25
25	13	10	34	25
26	12	9	32	25
27	15	9	25	25
28	13	10	46	25
29	15	10	43	25
30	13	10	27	25
Average of 30 runs	13.5	9.77	33.1	25.2

Table 5.4b. Comparison of zero entries[a]

Run	Zero entries				Total zero entries
	Storage area				
	1	2	3	4	
Worst-case	840	903	487	791	3021
1	823	908	526	775	3032
2	816	893	589	769	3067
3	853	900	551	733	3037
4	838	911	559	744	3052
5	828	899	584	728	3039
6	834	913	492	798	3037
7	795	901	617	768	3081
8	798	891	603	789	3081
9	784	890	648	756	3078
10	793	897	635	762	3087
11	848	904	466	808	3026
12	831	904	512	796	3043
13	802	893	602	784	3081
14	784	891	658	758	3091
15	833	904	480	815	3032
16	834	903	557	768	3062
17	829	893	568	766	3056
18	805	914	640	737	3096
19	851	923	516	769	3059
20	839	892	642	683	3056
21	838	911	594	707	3050
22	806	894	565	803	3068
23	835	893	543	766	3037
24	819	905	540	773	3037
25	816	905	564	783	3068
26	863	922	497	761	3043
27	844	897	624	723	3088
28	794	894	634	744	3069
29	825	887	532	808	3052
30	829	911	576	724	3040
Ave. of 30 runs	822.9	901.4	570.4	763.2	3058.2

[a]Total number of boxes:
size 1 = 995 size 3 = 1002
size 2 = 999 size 4 = 1004.

maximum contents of storage area 3 for the 30 random runs fall within the range from 24 to 46. This gives an average of 33.1 boxes, compared with 44 boxes of the worst-case permutation. The difference between 33.1 and 44 boxes corresponds to 25% increments for the demands on storage area 3. Based on the criterion of maximum contents, the worst-case permutation does place severe demands on the off-line storage space.

In Table 5.4b, the total zero entries of the 30 random runs range from 3,026 to 3,096 out of a total of 4,000 boxes. This yields an average of 3,058 boxes, compared with 3,021 boxes of the worst-case permutation. The total zero entries of all 30 random runs are consistently greater than those of the worst-case permutation. More operation time of moving the robot arm to and from the storage areas may be required for the worst-case permutation. Again, the worst-case permutation generates least efficient palletizing operations in terms of the number of directly loaded boxes. The worst-case permutation listed in Table 5.3 does yield a significant distribution transition sequence for the 20 box distributions. This sequence is thus used for the following simulations.

2. Multi-pallet packing (known distributions)

In this simulation, single-, double-, or triple-, and quadruple-pallet packing have been carried out. This refers to simultaneous loading of one, two, three and four pallets, respectively. Known distributions were applied. This means that the length of a distribution run

and the associated box proportion of each type can be determined before palletizing starts. Pallet patterns were changed only at the end of a distribution run and start of another new distribution run.

Tables 5.5 and 5.6 show the palletizing statistics of the total simulation for single-, double-, triple-, and quadruple-pallet packing. These summarize the overall simulation results of 4,000 boxes for the four experiments. The detailed palletizing statistics of each individual distribution run are presented in Appendix K.

Evaluation criteria of loading statistics and queues in storage areas are discussed separately in the subsections that follow.

a. Results of loading statistics Table 5.5 summarizes the total simulation results of loading statistics. The total number of boxes loaded onto the pallets is 4,000 boxes at the end of the simulation for all four pallet packing procedures.

Total palletizing time is reduced from 25.5 to 21.3 hours when the number of simultaneously loaded pallets increases from one to four. This corresponds to a 16.5% improvement in the total palletizing time. Furthermore, the total operation time that the robot moves to and from storage areas is reduced from 9.0, 5.7, 2.9 to 1.9 hours when the number of pallets increases from 1, 2, 3 to 4, respectively. Quadruple-pallet packing yields a 1.9-hour time in the storage area. This compares with 9 hours for the single-pallet packing. The difference of 7.1 hours represents 78.9% improvement in terms of the total operation time in storage areas.

Table 5.5. Loading statistics of multi-pallet packing

Statistics	Number of pallets			
	1	2	3	4
Total boxes loaded	4,000	4,000	4,000	4,000
Total simulation time (hours)	25.5	23.6	21.9	21.3
% improvement	-	7.5%	14.1%	16.5%
Total operation time in storage (hr)	9.0	5.7	2.9	1.9
Average cycle time (seconds)	22.9	21.2	19.7	19.2

b. Results of queue statistics Tables 5.6a through 5.6d summarize the simulation results of the queue statistics for off-line storage areas 1, 2, 3 and 4, respectively. Review of the data in these tables reveals that there is observable trend. Values of Average Contents, Average Waiting Time, Total Entries and Maximum Contents decrease and values of Zero Entries increase as the number of simultaneously loaded pallets increased. Storage area 3 shows the most dramatic change in statistical measures (see Table 5.6c). The average contents move from 7.19 boxes to only 0.05 boxes, and the average waiting time drops from 658.3 seconds to 3.8 seconds while the number of simultaneously loaded pallets increases from 1 to 4. The maximum contents of storage area 3 significantly drops from 44 boxes to only 5 boxes as the number of simultaneously loaded pallets increase. The improvements of maximum

Table 5.6a. Queue statistics of storage area 1

Statistics (storage area 1)	Number of pallets			
	1	2	3	4
Average contents	0.42	0.22	0.22	0.15
Average waiting time (seconds)	38.5	18.6	17.3	11.9
Total boxes generated	995	995	995	995
Total entries	155	85	57	40
Zero entries	840	910	938	955
Maximum contents	12	10	11	9

Table 5.6b. Queue statistics of storage area 2

Statistics (storage area 2)	Number of pallets			
	1	2	3	4
Average contents	0.16	0.08	0.06	0.05
Average waiting time (seconds)	14.4	6.4	5.0	3.9
Total boxes generated	999	999	999	999
Total entries	96	33	23	17
Zero entries	903	966	976	982
Maximum contents	10	9	8	7

Table 5.6c. Queue statistics of storage area 3

Statistics (storage area 3)	Number of pallets			
	1	2	3	4
Average contents	7.19	2.65	0.25	0.05
Average waiting time (seconds)	658.3	225.0	19.9	3.8
Total boxes generated	1002	1002	1002	1002
Total entries	515	354	102	27
Zero entries	487	648	900	975
Maximum contents	44	20	9	5

Table 5.6d. Queue statistics of storage area 4

Statistics (storage area 4)	Number of pallets			
	1	2	3	4
Average contents	1.09	0.95	0.86	0.69
Average waiting time (seconds)	100.1	80.3	67.6	53.1
Total boxes generated	1004	1004	1004	1004
Total entries	213	145	120	108
Zero entries	791	859	884	896
Maximum contents	25	23	21	19

contents for storage areas 1, 2 and 4 are not as dramatic.

Storage area 3 stores only boxes of size 2" x 2" x 2". The decreasing demands on storage space mean that the maximum required volume of storage area 3 is reduced from 352 cubic inches to only 40 cubic inches. This is an important result because of the limited work envelope that can be accessed by the robot. This gives an 88.6% improvement for the maximum demands on storage area 3. Since the requirements of storage areas are reduced, the frequency of storage area overflows can be reduced. This reduces the requirements for human intervention.

Consider the summation of zero entries of storage areas 1, 2, 3 and 4. They are as shown below.

Pallet packing	Total zero entries	% improvement
Single	3021	--
Double	3383	12.0%
Triple	3698	22.4%
Quadruple	3808	26.0%

While the number of pallets increases from 1 to 4, the total zero entries increase from 3021 to 3808 boxes. This corresponds to up to 26.0% improvements when compared with single-pallet packing. This indicates that less robot movements to and from storage areas are required. The results of total operation time in storage areas (see Table 5.5) reflect this improvement.

Based on the results of the queue statistics, the quadruple-pallet packing is again the best palletizing procedure in terms of the demands on storage space.

Indeed, statistical measures are improved while the number of simultaneously loaded pallets increases. Multi-pallet packing approach results in high efficient palletizing procedure, which requires less total palletizing time to load a fixed number of boxes, and reduces non-productive robot movements to and from storage areas. Also, multi-pallet packing requires less space of off-line storage areas. This turns out to be feasible to load boxes of many sizes with limited robot work envelope.

3. Look-ahead factors (unknown distributions)

Recall that the look-ahead factor (α) in equation 4.1 determines how long the look-ahead queue should be in terms of box volumes. In this simulation, this parameter is varied from 1 to 3 in increments of 0.5. This refers that box volumes in the look-ahead queue should be accumulated until they reach 1 to 3 times of a pallet's volume, respectively.

Single-pallet packing with unknown distributions is applied here. An unknown distribution means that information on the length of a distribution run and the associated box proportions is not available to the palletizing control program before palletizing starts. The worst-case permutation of 20 box distributions are also employed; however, these data are assumed to be unknown to the robot. The procedure of dynamic selection for a best match pallet pattern described in section D.2.b of Chapter IV is implemented in this simulation. This dynamic selection procedure is carried out whenever a pallet is full.

Tables 5.7 and 5.8 show the palletizing statistics for the total simulation with alternate look-ahead factors. These data summarize the overall simulation results of 4,000 boxes for each experiment. The detailed palletizing statistics of each individual distribution run are presented in Appendix K.

Simulation results of loading and queue statistics for each storage area are separately discussed in the subsections that follow.

a. Results of loading statistics Table 5.7 summarizes the loading statistics of five look-ahead factors, 1, 1.5, 2, 2.5 and 3. With the look-ahead factor of 3, there is a total of 3,985 out of 4,000 boxes loaded onto the pallets at the end of the simulation. This gives the maximum number of boxes loaded to the pallets in the five experiments. With a look-ahead factor of 1.5, 3,981 boxes have been loaded to the pallets. The difference between $\alpha = 1.5$ and $\alpha = 3$ is only 4 boxes.

Total palletizing time, total operation time in storage areas, and total zero entries for look-ahead factors 2, 2.5 and 3 are almost identical to one another.

Only 3,942 boxes are loaded to the pallets when the look-ahead factor is set to 1. With $\alpha = 1$, the total palletizing time, total operation time in storage areas and total zero entries are better than those with $\alpha = 2$, 2.5 and 3, but not as good as those with $\alpha = 1.5$.

With the look-ahead factor of 1.5, a minimum total palletizing time of 25.7 hours is achieved. This compares with 27.0 hours with

Table 5.7. Loading statistics of alternate look-ahead factors

Statistics	Look-ahead factor (α)				
	1	1.5	2	2.5	3
Total boxes loaded	3942	3981	3972	3972	3985
Total simulation time (hrs)	26.4	25.7	27.0	27.0	27.0
Total operation time in storage (hrs)	11.0	9.7	12.4	12.2	12.4
Average cycle time (sec)	23.8	23.2	24.3	24.3	24.3
Total zero entries	2778	2938	2672	2651	2631

$\alpha = 3$. This corresponds to 1.3 hours or 4.8% improvement. An α value of 1.5 yielded the minimum total operation time in storage areas of 9.7 hours. This compares with 12.4 hours with $\alpha = 3$. This is a 21.7% improvement. Finally, the procedure with $\alpha = 1.5$ generates the maximum number of total zero entries in all five experiments. This means that less robot movements to and from storage areas are required.

Based on the performance of the loading statistics, the look-ahead factor of 1.5 generates most efficient palletizing procedure.

b. Results of queue statistics Tables 5.8a through 5.8d summarize the queue statistics for storage areas 1, 2, 3 and 4, respectively. In the five experiments, average contents and average waiting time for the 1.5 look-ahead factor yield the best or the second best results for the four storage areas. With $\alpha = 1.5$, the average contents of

Table 5.8a. Queue statistics of storage area 1

Statistics (storage area 1)	Look-ahead factor (α)				
	1	1.5	2	2.5	3
Average contents	11.73	4.83	2.23	15.86	17.25
Average waiting time (seconds)	1121.3	450	218.0	1550.2	45.0
Total boxes generated	995	995	995	995	995
Total entries	464	433	232	606	403
Zero entries	531	562	763	389	592
Maximum contents	63	24	24	65	77
Current contents	58	13	0	26	0

Table 5.8b. Queue statistics of storage area 2

Statistics (storage area 2)	Look-ahead factor (α)				
	1	1.5	2	2.5	3
Average contents	0.85	0.60	6.10	2.61	9.72
Average waiting time (seconds)	80.7	55.9	593.9	254.2	944.8
Total boxes generated	999	999	999	999	999
Total entries	196	208	574	271	573
Zero entries	803	791	425	728	426
Maximum contents	20	15	35	25	74
Current contents	0	0	18	2	6

Table 5.8c. Queue statistics of storage area 3

Statistics (storage area 3)	Look-ahead factor (α)				
	1	1.5	2	2.5	3
Average contents	0.88	0.21	0.03	0.53	0.43
Average waiting time (seconds)	84.3	19.4	3.1	51.9	41.9
Total boxes generated	999	999	999	999	999
Total entries	150	132	49	189	119
Zero entries	849	867	950	810	880
Maximum contents	22	7	5	15	15
Current contents	0	6	0	0	9

Table 5.8d. Queue statistics of storage area 4

Statistics (storage area 4)	Look-ahead factor (α)				
	1	1.5	2	2.5	3
Average contents	2.00	0.78	2.72	1.40	0.66
Average waiting time (seconds)	188.9	71.9	262.6	135.3	64.1
Total boxes generated	1007	1007	1007	1007	1007
Total entries	412	289	473	283	274
Zero entries	595	718	534	724	733
Maximum contents	28	14	24	20	15
Current contents	0	0	10	0	0

storage areas 2, 3 and 4 are only 0.6, 0.21 and 0.78 boxes, respectively. The average contents of storage area 1 are 4.83 boxes, compared with a best value of 2.23 boxes when $\alpha = 2$.

The overall averages of "average contents" and "average waiting time" for the four storage areas are listed in Table 5.9. In this table, the overall average of "average contents" equals

$$\frac{1}{4} \sum_{i=1}^{4} \text{(average contents of storage area i)}.$$

For look-ahead factor of 1, the overall average of "average contents" is

$$\frac{1}{4} (11.73 + 0.85 + 0.88 + 2.00) = 3.86.$$

Table 5.9. The overall average statistics

Overall average statistics	Look-ahead factor (α)				
	1	1.5	2	2.5	3
Ave. "average contents"	3.86	1.60	2.77	5.10	6.90
Ave. "average waiting time" (seconds)	368.8	149.3	269.4	497.9	273.9

From the above table, notice that the look-ahead factor of 1.5 gives minimum overall averages of numbers of boxes residing in the storage areas and minimum waiting time per box in the storage area. These values are 1.60 boxes and 149.3 seconds, respectively. No values for the re-

maining look-ahead factors are even close to these two numbers. Moreover, the look-ahead factor of 1.5 generates smallest numbers of maximum contents for storage areas 1, 2 and 4. The maximum contents of storage area 3 is only 7 boxes. This compares with a "best" maximum contents value of 5 boxes when $\alpha = 2$. The difference is only 2 boxes. Thus, the look-ahead factor of 1.5 gives the palletizing procedure that requires least space of off-line storage area. Based on the consideration of demands on storage space, the look-ahead factor of 1.5 generates best palletizing procedure.

From the overall comparison of palletizing efficiency and demands on storage space, the look-ahead factor of 1.5 generates the best loading procedure which requires least palletizing time to complete the placement of 4,000 boxes. It requires the least nonproductive robot movements to and from storage areas, and the least off-line storage space. Surprisingly, the larger look-ahead factors did not generate better palletizing performance. This makes the dynamic selection procedure even more feasible and practical for industrial applications. Since the observed length of a look-ahead queue is only 1.5 times of a pallet volume, the length of most industrial conveyors may be sufficient to contain the boxes to be observed at a time.

The good performance of the 1.5 look-ahead factor over other larger look-ahead factors may be explained by comparing the pallet patterns selected during the palletizing process. When a pallet is fully loaded, the pallet pattern is dynamically selected from the 20 different pallet

patterns stored on the disk. The pallet patterns selected for the first three distribution runs (distributions 1, 18 and 4) in the comparison of look-ahead factors 1.5 vs. 3 are shown in sequence in Table 5.10.

From Table 5.10, the pallet patterns selected for both look-ahead factors 1.5 and 3 are almost identical except toward the end of each distribution run. For the second distribution run (distribution 18), three consecutive pallet pattern #18s are selected for $\alpha = 1.5$. However, only two consecutive pallet pattern #18s are selected for $\alpha = 3$. The second distribution run (distribution 18) generates 200 boxes of type 1 (see Table 5.1). A full pallet load for pallet pattern 18 consists of 64 boxes of type 1 (see Table 5.2). For $\alpha = 3$, after 128 boxes of type 1 are loaded to two pallet pattern #18s, there are 72 boxes left in distribution 18 to be loaded. These 72 boxes consist of only 1.125 times of the pallet's volume (64 cu. in.). Since the look-ahead factor of 3 is used, the boxes in distribution 4 (the third distribution run) must be included in the look-ahead queue until the cumulative box volumes reach 3 times the pallet's volume. This forces the procedure to select the pallet pattern 5 early as opposed to pallet pattern #18.

Therefore, the maximum contents of storage 1 are dramatically increased to 62 boxes for $\alpha = 3$, compared with only 6 boxes for $\alpha = 1.5$. This is because pallet pattern 5 cannot absorb type 1 boxes as quickly as pallet pattern 18. For $\alpha = 3$, 156 out of a total of 200 boxes are loaded onto the pallets. At the end of the second distribution run, there are 62 boxes of type 1 residing in storage area 1. This result

Table 5.10. Pallet patterns selected for the first three distribution runs

Distribution run	Distribution[a] number	Pallet[b] loaded	Pallet pattern selected	
			$\alpha = 1.5$[c]	$\alpha = 3$
1	1	1	1	1
		2	1	1
		3	1	1
		4	1	1
		5	1	1
		6	1	1
		7	1	1
		8	1	1
		9	1	1
		10	1	1
		11	2	1
		12	1	1
		13	1	1
		14	1	1
		15	1	3
		16	8	
2	18	1	18	18
		2	18	18
		3	18	5
		4	5	5

[a] Distribution 1 is simulated first (the first distribution run), distribution 18 is the second, and so on.

[b] For each distribution run, 200 boxes are generated. The "pallet loaded" column shows the first full pallet loaded, the second full pallet, and so on.

[c] For $\alpha = 1.5$, the first "pallet loaded" uses pallet pattern #1, and the 11th "pallet loaded" uses pallet pattern #2, etc.

Table 5.10. continued

Distribution run	Distribution number	Pallet loaded	Pallet pattern selected $\alpha = 1.5$	Pallet pattern selected $\alpha = 3$
3	4	1	4	4
		2	4	4
		3	4	2
		4	2	4
		5	4	4
		6	4	4
		7	4	4
		8	4	4
		9	4	4
		10	4	4
		11	4	4
		12	4	4
		13	4	1
		14	8	9

thus degrades the overall performance of the total simulation. The palletizing and queue statistics of distribution 18 for look-ahead factors 1.5 and 3 are presented in Table 5.11. The resulting statistics for each individual distribution run can also be found in Appendix K.

4. Comparison of known and unknown box distributions

With unknown distributions, it is assumed that the information of incoming box size proportions and length of a distribution run is completely not available before palletizing starts. In section D.2.b of Chapter IV, the procedure of dynamically selecting a "best match" pal-

Table 5.11. Palletizing and queue statistics of distribution 18

Statistics	Look-ahead factor $\alpha = 1.5$	Look-ahead factor $\alpha = 3$
Total boxes generated	200	200
Total boxes loaded	198	156
Total palletizing time (sec)	3,483	3,620
Operation time in storage (sec)	150	1,112
For storage area 1:		
Average contents	0.08	7.5
Average waiting time (sec)	1.5	135.8
Maximum contents	6	62
Total entries	6	64
Zero entries	194	136
Current contents	6	62

let pattern according to the boxes in the look-ahead queue has been described. To evaluate the performance and responsive capability of this dynamic selection procedure, the palletizing statistics of the previous two simulation results are compared with each other. The simulation results of single-pallet packing with known box distributions are employed as a basis of comparison. These compared with the simulation results for the look-ahead factor of 1.5 with unknown box distributions. These palletizing statistics are extracted from Tables 5.5 and 5.6a-d and Table 5.7 and 5.8a-d, respectively, and presented in Tables 5.12 and 5.13a-d.

Comparisons of the loading and the queue statistics are separately discussed in the following subsections.

Table 5.12. Comparison of loading statistics

Measures \ Situation	Known distribution	Unknown[a] distribution	Difference	% of difference
Total boxes loaded	4,000	3,981	19	0.5%
Total simulation time (hr)	25.5	25.7	0.2	0.8%
Total operation time in storage (hr)	9.0	9.7	0.7	7.8%
Average cycle time (sec)	22.9	23.2	0.3	1.3%
Total zero entries	3,021	2,938	83	2.7%

[a]Look-ahead factor = 1.5.

a. Comparison of loading statistics Table 5.12 summarizes the loading statistics of the known (single-pallet packing) and unknown (the look-ahead factor of 1.5) distributions. From Table 5.12, notice that the percentage differences of all statistical measures between known and unknown distributions range from 0.5% to a maximum of 7.8%. The total palletizing time for the known distributions is 25.5 hours, compared with 25.7 hours for the unknown distributions. The difference is only 0.2 hours or 0.8%. The total operation time in storage area for the unknown distributions is slightly longer than the known distributions. This gives a difference of 0.7 hours or 7.8%. This indicates that the performance difference in terms of the loading statistics between known and unknown box size distributions is not significant.

Table 5.13a. Comparison of queue statistics (storage area 1)

Measures \ Situation	Known distribution	Unknown distribution
Average contents	0.42	4.83
Average waiting time (seconds)	38.5	450
Total boxes generated	995	995
Total entries	155	433
Zero entries	840	562
Maximum contents	12	24
Current contents	0	13

Table 5.13b. Comparison of queue statistics (storage area 2)

Measures \ Situation	Known distribution	Unknown distribution
Average contents	0.16	0.60
Average waiting time (seconds)	14.4	55.9
Total boxes generated	999	999
Total entries	96	208
Zero entries	903	791
Maximum contents	10	15
Current contents	0	0

Table 5.13c. Comparison of queue statistics (storage area 3)

Measures \ Situation	Known distribution	Unknown distribution
Average contents	7.19	0.21
Average waiting time (seconds)	658.3	19.4
Total boxes generated	1002	999
Total entries	515	132
Zero entries	487	867
Maximum contents	44	7
Current contents	0	6

Table 5.13d. Comparison of queue statistics (storage area 4)

Measures \ Situation	Known distribution	Unknown distributions
Average contents	1.1	0.78
Average waiting time (seconds)	100.1	71.9
Total boxes generated	1004	1007
Total entries	213	289
Zero entries	791	718
Maximum contents	25	14
Current contents	0	0

Based on the loading statistics, the dynamic selection procedure for unkown distributions performed equally well as the pallet packing with known distributions. The dynamic selection procedure with a look-ahead factor of 1.5 can yield a palletizing procedure that is equally efficient with those of known box size distributions.

b. Comparison of queue statistics Tables 5.13a through 5.13d summarize the queue statistics of known (single-pallet packing) and unknown (the look-ahead factor of 1.5) distributions. There is an interesting trend in the queue statistics in Tables 5.13a-d. With known distributions, the pallet packing process tends to minimize the demands on storage areas 1 and 2. The palletizing procedure with unknown distributions tends to minimize the demands on storage areas 3 and 4. For storage area 1, the average contents and average waiting time for the known distributions are 0.42 boxes and 38.5 seconds per box, compared with 4.83 boxes and 450 seconds for the unknown distributions, respectively. In contrast, for storage 3, the average contents and average waiting time for the unknown distributions are only 0.21 boxes and 19.4 seconds per box, compared with 7.19 boxes and 658.3 seconds for the known distributions, respectively. These differences are extremely significant.

By examining all pallet patterns selected during the palletizing process for distribution 2, the reason for difference in queue statistics of storage area 3 for known vs. unknown distributions becomes apparent.

For known distributions, pallet pattern 2 is applied for the entire

distribution run of 200 boxes. For unknown distributions, pallet patterns are dynamically selected among the 20 pre-stored pallet patterns according to incoming box types. A full pallet load for pallet pattern 2 comprises four type 2 (1 x 2 x 1) and seven type 3 (2 x 2 x 2) boxes (see Table 5.2). Also, the random number generator generates 66 boxes of type 2 (33.3%) and 134 boxes of type 3 (66.7%) for distribution 2 (see Table 5.1).

If pallet pattern 2 is used for the entire distribution run of 200 boxes, the maximum number of full pallets loaded is equal to

$$\min\{[\frac{66}{4}],\ [\frac{134}{7}]\} = 16.$$

There are 16 full pallets, and each can be loaded with seven boxes of type 3. At the end of the distribution run, a partially loaded pallet may be placed with seven boxes of type 3. Therefore, at most 119 boxes of type 3 can be loaded onto the pallets. There are 134 boxes of type 3 generated in distribution 2. At least 15 boxes must be placed in storage area 3 since no sufficient boxes of type 2 exist to complete the pallet load for pattern #2. This means that the maximum and current contents for storage area 3 are at least 15 if pallet pattern 2 is used for the entire distribution run.

For unknown distributions, pallet patterns are dynamically selected according to incoming box types. Note that toward the end of the distribution run, the procedure selects pallet pattern 14 (and pallet patterns 1 and 3). This is shown in Table 5.14.

Table 5.14. Pallet patterns selected for distribution 2

Distribution number	Pallet loaded (in sequence)	Pallet pattern selected	
		Known dist.	Unknown dist.
2	1	2	2
	2	2	2
	3	2	14
	4	2	2
	5	2	2
	6	2	2
	7	2	2
	8	2	2
	9	2	2
	10	2	2
	11	2	2
	12	2	2
	13	2	2
	14	2	2
	15	2	2
	16	2	2
	17		14
	18		2
	19		1
	20		3

A full pallet load for pattern #14 consists of eight type 3 boxes. Therefore, type 3 boxes picked up from the infeeding conveyor can be directly loaded onto the pallet pattern 14 without waiting for other type 2 boxes. This dynamic selection procedure for a "best match" pallet pattern is therefore more efficient.

The queue statistics for storage area 3 in the comparison of known vs. unknown distributions for distribution 2 are presented in Table 5.15. The queue statistics for other individual distribution runs are presented in Appendix K. For known distributions, the maximum contents of storage area 3 are 19 boxes. This compares with 7 boxes for unknown distributions. At the end of the distribution run, 15 boxes ("current contents" in Table 5.15) of type 3 reside in storage are 3 for known distributions. In contrast, there are only 2 boxes of type 3 left in storage

Table 5.15. Queue statistics of storage area 3 for distribution 2

Statistics (storage area 3)	Known distributions	Unknown distributions
Total boxes generated	134	134
Average contents	8.0	1.9
Average waiting time (seconds)	291.2	44.2
Maximum contents	19	7
Total entries	78	49
Zero entries	56	85
Current entries	15	2

area 3 for the unknown distribution situation. Therefore, the maximum contents keeps accumulating in the subsequent distribution runs for known distributions.

It is difficult to determine which palletizing procedure is more desirable in terms of the demands on storage areas. The maximum contents, which determine required space in off-line storage areas, are used as the criterion for the final decision. The maximum contents and their corresponding required space in cubic inches are summarized in Table 5.16.

Table 5.16. Maximum contents and required storage space

Storage (volume/box)	Maximum contents		Space (cu. in.)	
	Known dist.	Unknown dist.	Known dist.	Unknown dist.
Storage 1 (1 x 1 x 1)	12	24	12	24
Storage 2 (1 x 2 x 1)	10	15	20	30
Storage 3 (2 x 2 x 2)	44	7	352	56
Storage 4 (2 x 3 x 1)	25	14	150	84
Total	91	60	534	194

Again, the palletizing procedure for known box size distributions favors the maximum contents of storage areas 1 and 2. In contrast, the palletizing procedure for the unknown distributions generates smaller maximum contents requirements for storage areas 3 and 4, especially for storage area 3. The procedure for the unknown distributions dramatically drops the maximum contents of storage area 3 from 44 to only 7 boxes.

Storage areas 1, 2, 3 and 4 store boxes of sizes 1" x 1" x 1", 1" x 2" x 1", 2" x 2" x 2" and 2" x 3" x 1". This refers to 1, 2, 8 and 6 cubic inches per box, respectively. The total maximum contents of known and unknown distributions are 91 and 60 boxes, respectively. In terms of total maximum contents, the palletizing procedure for unknown distributions is better than the procedure for known distributions. In addition, the total required storage spaces for known and unknown distributions are 524 and 194 cubic inches, respectively. In terms of total required storage space, the palletizing procedure for unknown distributions is better. Based on the queue statistics, the palletizing procedure for unknown box size distribution places less demand on off-line storage space.

From the overall comparisons of the loading and the queue statistics, the dynamic selection procedure for a "best match" pallet pattern performed as well as the pallet loading process for known distributions. It performed even better than the palletizing procedure for known distributions in terms of demands on storage space. Since this dynamic selection procedure can perform the palletizing procedure as if

the box size distributions are known, it is feasible to apply it for the situation in which box size proportions and length of a distribution run are not determinable before the palletizing starts.

5. Summary

The objectives of this simulation have been to evaluate the feasibility and performance of the robotic palletizing system, examine the procedure of multi-pallet packing, determine the best look-ahead factor for the unknown distribution situation, and evaluate the dynamic selection procedure for a "best match" pallet pattern.

With multi-pallet packing, more boxes can be directly loaded onto the pallets, and nonproductive palletizing movements to and from storage areas can be minimized. The palletizing efficiency is gradually increased, and demands on storage areas are continuously reduced as the number of simultaneously loaded pallet increases.

The observed look-ahead queue length in terms of box volume has been found to be the best when it is 1.5 times of the pallet's volume. With a look-ahead factor of 1.5, the dynamic selection procedure for a "best match" pallet pattern performs most efficient palletizing operations and requires least space in off-line storage areas. The dynamic selection procedure for unknown distributions performed as well as the single-pallet packing with known distributions.

With the integration of dynamic pallet patterns and multi-pallet packing systems, it has been found that the robotic palletizing system generates efficient loading operations, and makes the demands on off-line storage space that are at feasible levels.

This palletizing simulation has employed a miniature physical simulator. All developed palletizing softwares and systems are directly extendable to full-size equipment for actual industrial applications.

VI. CONCLUSIONS

The developed mixed 0-1 integer programming algorithm is an exact procedure to solve the three-dimensional pallet loading problem. It generates the required numbers for each type of box, and the xyz-coordinate placement location for every box on the pallet. The heuristic dynamic programming procedure generates a good solution with less computation time. No human intervention is necessary for an optimal pallet pattern. These two algorithms have eliminated the requirements of finding an optimal solution from millions of possible combinations.

It has been reported that one of Nabisco Biscuit Group's highrise storage centers saves $2 million a year in freight cost alone by shipping 10-15% more merchandise per truckload [98]. Since the algorithm will maximize the use of pallet space, total number of required pallets will be reduced and truckload box capacity could be improved.

Use of industrial robots for palletizing releases people from tedious and fatiguing work. It helps increase productivity, reduce human packing errors, save labor and prevent accidents. Most important, robot palletization provides flexibility; the ability to accommodate frequent changes of sophisticated pallet patterns without changing equipment. Due to the use of automatic palletizing system, one of Campbell Soup Company's plants obtained annual savings of over $500,000 in labor costs and 40% better throughput [35].

Based on the simulation results, the designed robotic palletizing procedure has performed efficiently for the pallet loading with multiple

box sizes. With or without advanced information of box proportions, pallet patterns can be dynamically altered according to the incoming box distributions. The developed mathematical algorithms and robotic palletizing systems provide efficiency and flexibility for both warehousing and manufacturing industries involving complex pallet packing requirements. Mixed, as opposed to identical, cartons have been addressed in this research. This is a significant extension to existing automated systems.

VII. BIBLIOGRAPHY

1. Abair, D. W. "Modern Solutions to Old Problems-Palletizing with Industrial Robots." Proceedings of Robots 8: Applications for Today. Vol. 1. Robotics International of SME, Detroit, Michigan, June 1984.

2. Adamowicz, M. and A. Albano. "A Two-Stage Solution of the Cutting-Stock Problem." In Information Processing 71. New York: North-Holland, 1972.

3. Adamowicz, M. and A. Albano. "A Solution of the Rectangular Cutting-Stock Problem." IEEE Transactions on Systems, Man, and Cybernetics, SMC-6, No. 4 (1976), 302-310.

4. Albano, A. and R. Orsini. "A Heuristic Solution of the Rectangular Cutting-Stock Problem." The Computer Journal, 23, No. 4 (1980), 338-343.

5. Albano, A. and G. Sapuppo. "Optimal Allocation of Two-Dimensional Irregular Shapes Using Heuristic Search Methods." IEEE Transactions on System, Man, and Cybernetics, SMC-10, No. 5 (1980), 242-248.

6. Baker, B. S., D. J. Brown and H. P. Katseff. "A 5/4 Algorithm for Two-Dimensional Packing." Journal of Algorithms, 2, No. 4 (1981), 348-368.

7. Barnes, F. W. "Packing the Maximum Number of m x n Tiles in a Large p x q Rectangle." Discrete Mathematics, 26, No. 2 (1979), 93-100.

8. Basin, S. L. "Generalized Fibonacci Sequences and Squared Rectangles." American Mathematical Monthly, 70, No. 4 (1963), 372-379.

9. Beasley, J. E. "Algorithms for Unconstrained Two-Dimensional Guillotine Cutting." Journal of the Operational Research Society, 36, No. 4 (1985), 297-306.

10. Beasley, J. E. "An Algorithm for the Two-Dimensional Assortment Problem." European Journal of Operational Research, 19, No. 2 (1985), 253-261.

11. Beasley, J. E. "An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure." Operations Research, 33, No. 1 (1985), 49-64.

12. Beasley, J. E. "Bounds for Two-Dimensional Cutting." Journal of the Operational Research Society, 36, No. 1 (1985), 71-74.

13. Biró, M. and E. Boros. "Network Flows and Non-Guillotine Cutting Patterns." European Journal of Operational Research, 16, No. 2 (1984), 215-221.

14. Bischoff, E. and W. B. Dowsland. "An Application of the Micro to Product Design and Distribution." Journal of the Operational Research Society, 33, No. 3 (1982), 271-280.

15. Brown, A. R. Optimum Packing and Depletion. New York: American Elsevier, 1971.

16. Brown, D. J. "An Improved BL Lower Bound." Information Processing Letters, 11, No. 1 (1980), 37-39.

17. Brualdi, R. A. and T. H. Foregger. "Packing Boxes with Harmonic Bricks." Journal of Combinatorial Theory, 17, No. 2 (1974), 81-114.

18. Carpenter, H. and W. B. Dowsland. "Practical Considerations of the Pallet-Loading Problem." Journal of the Operational Research Society, 36, No. 6 (1985), 489-497.

19. Cheng, M. H. R. and A. W. Pila. "Maximized Utilization of Surface Areas with Defects by the Dynamic Programming Approach." ISA Transactions, 17, No. 4 (1978), 61-69.

20. Christofides, N. and C. Whitlock. "An Algorithm for Two-Dimensional Cutting Problems." Operations Research, 25, No. 1 (1977), 30-44.

21. Chung, F. R. K., M. R. Garey and D. S. Johnson. "On Packing Two-Dimension Bins." SIAM Journal on Algebraic and Discrete Methods, 11, No. 1 (1980), 37-39.

22. Chung, F. R. K., E. N. Gilbert, and R. L. Graham. "Tiling Rectangles with Rectangles." Mathematics Magazine, 55, No. 10 (1979), 286-291.

23. Coffman, E. G., Jr., M. R. Garey, D. S. Johnson and R. E. Tarjan. "Performance Bounds for Level-Oriented Two-Dimensional Packing Algorithms." SIAM Journal on Computing, 19, No. 4 (1980), 808-826.

24. Cooper, M. W. and K. Farhangian. "A Dynamic Programming Algorithm for Multiple-Choice Constraints." Computers and Mathematics with Applications, 10, No. 3 (1984), 279-282.

25. Côté, G. and M. A. Laughton. "Large-Scale Mixed Integer Programming: Benders-Type Heuristics." European Journal of Operational Research, 16, No. 3 (1984), 327-333.

26. Cotter, S. M. and B. G. Batchelor. "Visual Monitoring of Palletizing and Packing." Proceedings of the 3rd International Conference on Robot Vision and Sensory Controls, Cambridge, Massachusetts, November 1983.

27. Davis, R. E., D. A. Kendrick and M. Weitzman. "A Branch-and-Bound Algorithm for Zero-One Mixed Integer Programming Problems." Operations Research, 19, No. 4 (1971), 1036-1044.

28. Dowsland, K. A. "Determining an Upper Bound for a Class of Rectangular Problems." Computers and Operations Research, 12, No. 2 (1985), 201-205.

29. Dowsland, K. A. "The Three-Dimensional Pallet Chart: An Analysis of the Factors Affecting the Set of Feasible Layouts for a Class of Two-Dimensional Packing Problems." Journal of the Operational Research Society, 35, No. 10 (1984), 895-905.

30. Dowsland, W. B. "Two and Three Dimensional Packing Problems and Solution Methods." New Zealand Operational Research, 13, No. 1 (1985), 1-18.

31. Farley, A. "Trim-Loss Pattern Rearrangement and Its Relevance to the Flat-Glass Industry." European Journal of Operational Research, 14, No. 4 (1983), 386-392.

32. Farley, A. "A Note on Modifying a Two-Dimensional Trim-Loss Algorithm to Deal with Cutting Restrictions." European Journal of Operational Research, 14, No. 4 (1983), 393-395.

33. Fleming, J. R. "The Development and Implementation of a Pallet Loading Algorithm with a Physical Robotic Model." M.S. Thesis. Iowa State University, Ames, Iowa, 1986.

34. Fleming, J. R., E. M. Malstrom and H. D. Meeks. "A Robotic Pallet-Loading Algorithm for Dynamic Carton Distributions." Proceedings of the 7th International Conference on Automation in Warehousing, San Francisco, California, December 1986.

35. "Flexible Palletizing Cuts Costs, Improves Service." Modern Materials Handling, 41, No. 2 (1986), 71-73.

36. Garey, M. R. and D. S. Johnson. "Approximation Algorithms for Bin Packing Problems: A Survey." In Analysis and Design of Algorithms in Combinatorial Optimization. Ed. G. Ausiello and M. Lucertine. CISM Courses and Lectures No. 266. International Center for Mechanical Sciences, Italy, 1981.

37. Garey, M. R. and D. S. Johnson. Computer and Intractability. New York: W. H. Freeman, 1979.

38. George, J. A. and D. F. Robinson. "A Heuristic for Packing Boxes into a Container." Computers and Operations Research, 7, No. 3 (1980), 147-156.

39. Gilmore, P. C. and R. E. Gomory. "A Linear Programming Approach to the Cutting-Stock Problem." Operations Research, 9, No. 6 (1961), 849-859.

40. Gilmore, P. C. and R. E. Gomory. "Multistage Cutting Stock Problems for Two and More Dimensions." Operations Research, 13, No. 1 (1965), 94-120.

41. Gilmore, P. C. and R. E. Gomory. "The Theory and Computation of Knapsack Function." Operations Research, 14, No. 6 (1966), 1045-1074.

42. Golan, I. "Performance Bounds for Orthogonal Oriented Two-Dimensional Packing Algorithms." SIAM Journal on Computing, 10, No. 3 (1981), 571-582.

43. Golden, B. L. "Approaches to the Cutting Stock Problem." AIIE Transactions, 8, No. 2 (1976), 265-274.

44. Goto, T. and K. Takeyasu. "Compact Packaging by Robot with Tactile Sensors." Proceedings of the 2nd International Symposium on Industrial Robots. IIT Research Institute, Chicago, Illinois, May 1972.

45. Grab, E. "Robot Palletizing Center (RPC)." Proceedings of the 14th International Symposium on Industrial Robots, and 7th International Conference on Industrial Robot Technology, Gothenburg, Sweden, October 1984.

46. Graham, R. L. "Fault-Free Tilings of Rectangles." In The Mathematical Gardner. Ed. D. A. Klarner. Belmont, California: Wadsworth International, 1981.

47. Gupta, A. D. "Operations Research Models for Design of Palletisation System." Journal of Institution of Engineers (India), Mechanical Engineering Division, 57, Pt. ME 4 (1977), 183-185.

48. Gupta, O. K. and A. Ravindran. "Branch and Bound Experiments in Convex Nonlinear Integer Programming." Management Science, 31, No. 12 (1985), 1533-1546.

49. Haessler, R. W. "A Note on Computational Modifications to the Gilmore-Gomory Cutting Stock Algorithm." Operations Research, 28, No. 4 (1980), 1001-1005.

50. Hahn, S. G. "On the Optimal Cutting of Defective Sheets." Operations Research, 16, No. 6 (1968), 1100-1115.

51. Haims, M. J. and H. Freeman. "A Multistage Solution of the Template-Layout Problem." IEEE Transactions on Systems Science and Cybernetics, SSC-6, No. 2 (1970), 145-151.

52. Healy, V. C., Jr. "Multiple Choice Programming." Operations Research, 12, No. 1 (1964), 122-138.

53. Herz, J. C. "Recursive Computational Procedure for Two-Dimensional Stock Cutting." IBM Journal of Research and Development, 16, No. 5 (1972), 462-469.

54. Hilbert, D. and P. Vortrag. Mathematische Problem: Problem 18. Nachr. Gesellsch. Wiss. Göttingen, Mathem.-Physik. Klasse, 1900.

55. Hillier, F. S. and G. J. Lieberman. Operations Research. San Francisco: Holden-Day Inc., 1974.

56. Hinxman, A. I. "The Trim-Loss and Assortment Problems: A Survey." European Journal of Operational Research, 5, No. 1 (1980), 8-18.

57. Hodgson, T. J. "A Combined Approach to the Pallet Loading Problem." IIE Transactions, 14, No. 3 (1982), 175-182.

58. Hodgson, T. J., D. S. Hughes and L. A. Martin-Vega. "A Note on a Combined Approach to the Pallet Loading Problem." IIE Transactions, 15, No. 3 (1983), 268-271.

59. Hoffman, D. G. "Packing Problems and Inequalities." In The Mathematical Gardner. Ed. D. A. Klarner. Belmont, California: Wadsworth International, 1981.

60. Israni, S. and J. Sanders. "Two-Dimensional Cutting Stock Problem Research: A Review and a New Rectangular Layout Algorithm." Journal of Manufacturing Systems, 1, No. 2 (1982), 169-182.

61. Jeroslow, R. G. and J. K. Lowe. "Experimental Results on the New Techniques for Integer Programming Formulations." Journal of Operations Research Society, 36, No. 5 (1985), 393-403.

62. Kershner, R. B. "On Paving the Plane." American Mathematical Monthly, 75, No. 8 (1968), 839-844.

63. Kochenberger, G. A. and V. H. Richard. "A Simple, All Primal Branch and Bound Approach to Pure and Mixed Integer Binary Problems." Operations Research Letters, 1, No. 5 (1982), 182-185.

64. Kulick, A. "Interlocking Pallet Pattern Simulation Program." Industrial Engineering, 14, No. 9 (1982), 22-24.

65. Lane, A. H. and A. G. Doig. "An Automatic Method of Solving Discrete Programming Problems." Econometrica, 28, No. 3 (1960), 497-520.

66. Lindkvist, R. G. T. Handbook of Materials Handling. West Sussex, England: Ellis Horwood Limited, 1985.

67. Marasinghe, M. G. Statistical Applications of Digital Computers. Supplementary Notes for Statistics 480. Department of Statistics, Iowa State University, Ames, Iowa, Fall 1984.

68. Martin, R. K. and L. Schrage. "Subset Coefficient Reduction Cuts for 0/1 Mixed-Integer Programming." Operations Research, 33, No. 3 (1985), 505-526.

69. Martin, R. K., D. J. Sweeney and M. E. Doherty. "The Reduced Cost Branch and Bound Algorithm for Mixed Integer Programming." Computers and Operations Research, 12, No. 2 (1985), 139-149.

70. Mathematical Programming System Extended (MPSX), Mixed Integer Programming (MIP). Program Number 5734-XM4. New York: IBM World Trade Corporation, 1971.

71. Miller, R. K. Robots in Industry. Madison, Georgia: SEAI Institute, Center for Robotic Applications, 1984.

72. Mitra, G. "Investigation of Some Branch and Bound Strategies for the Solution of Mixed Integer Linear Programming." Mathematical Programming, 4, No. 2 (1973), 155-170.

73. Ohtake, Y. and N. Nishida. "A Branch-and-Bound Algorithm for 0-1 Parametric Mixed Integer Programming." Operations Research Letters, 4, No. 1 (1985), 41-45.

74. Page, E. "A Note on a Two-Dimensional Dynamic Programming Problem." Operational Research Quarterly, 26, No. 2 (1975), 321-324.

75. "Palletizing and Unitizing: How Robots Stack Up." Modern Materials Handling, 39, No. 10 (1984), 59-62.

76. Rehg, J. Introduction to Robotics: A System Approach. Englewood Cliffs, N.J.: Prentice-Hall, 1985.

77. Roberts, S. A. "Application of Heuristic Techniques to the Cutting-Stock Problem for Worktops." Journal of the Operational Research Society, 35, No. 5 (1984), 369-377.

78. "Robot Picks, Counts, and Palletizes Parts." Modern Materials Handling, 39, No. 10 (1984), 64-65.

79. Robotics Reference and Applications Manual: MiniMover-5. Mountain View, California: Microbot, Inc., 1981.

80. "Robots: A Flexible Solution for Tough Unitizing Jobs." Material Handling Engineering, 38, No. 9 (1983), 36-40.

81. Ruthedge, R. W. "A Simplex Method for Zero-One Mixed Integer Linear Programs." Journal of Mathematical Analysis and Applications, 18, No. 2 (1967), 377-390.

82. Salzer, J. J. "Order Selection to Conveyors with Voice Encoding on Advanced Distribution Warehouse Incorporating a High Degree of Mechanisation and Automation." Proceedings of the 2nd International Conference on Automation in Warehousing, University of Keele, England, March 1977.

83. Sandhu, H. S. Hands-On-Introduction to Robotics, The Manual for the XR-Series Robots. Champaign, Illinois: Rhino Robots, 1983.

84. Schiwarov, N. St. and K. Iv. Yanakiev. "Mechanical Handling System for Automatic Quality Grading and Robotized Wall Tile Palletization." Proceedings of the 6th International Conference on Automation in Warehousing, Atlanta, Georgia, December 1983.

85. Schriber, T. J. Simulation Using GPSS. New York: John Wiley & Sons, Inc., 1974.

86. Shaphinpoor, M. "The Exact Inverse Kinematics Solutions for the Rhino XR-2 Robot Manipulator." Robotics Age, 7, No. 8 (1985), 6-14.

87. Smith, A. and P. de Cani. "An Algorithm to Optimize the Layout of Boxes in Pallet." Journal of the Operational Research Society, 31, No. 7 (1980), 573-578.

88. The Specifications and Applications of Industrial Robots in Japan: 1984. Tokyo: Japan Industrial Robot Association, 1984.

89. Stauffer, R. N. "Palletizing/Depalletizing: Robots Make It Easy." Robotics Today, 6, No. 1 (1984), 43-46.

90. Steudel, H. J. "The Right Pallet Cuts Costs." Industrial Engineering, 9, No. 2 (1977), 14-18.

91. Steudel, H. J. "Generating Pallet Loading Patterns: A Special Case of the Two-Dimensional Cutting Stock Problem." Management Science, 25, No. 10 (1979), 997-1004.

92. Steudel, H. J. "Generating Pallet Loading Patterns with Considerations of Item Stacking on End and Side Surfaces." Journal of Manufacturing Systems, 3, No. 2 (1984), 135-143.

93. Tanchoco, J. M. A. and M. H. Agee. "Plan Unit Loads to Interact with All Components of Warehouse System." Industrial Engineering, 13, No. 6 (1981), 36-47.

94. Thesen, A. Computer Methods in Operations Research. New York: Academic Press, 1978.

95. Tsai, D. M. The Use of a Robotic Manipulator to Unitize Pallet Loads in a Warehouse Distribution System. M.S. Thesis. Iowa State University, Ames, Iowa, 1984.

96. Tsai, D. M., E. M. Malstrom and H. D. Meeks. "Robotic Unitization of Pallet Loads." Proceedings of the Fall Industrial Engineering Conference, Institute of Industrial Engineers, Chicago, Illinois, December 1985.

97. Tsai, D. M., E. M. Malstrom and H. D. Meeks. "Modeling and Analysis of a Robotic Palletizing Cell." Working Paper #86444, Engineering Research Institute, Iowa State University, Ames, Iowa, May, 1986.

98. "Updating Your Distribution? Update Your Material Handling System First." Material Handling Engineering, 39, No. 4 (1984), 46-52.

99. Wang, P. Y. "Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems." Operations Research, 31, No. 3 (1983), 573-586.

100. "What's New in Palletizers." Modern Materials Handling, 40, No. 3 (1985), 80-82.

101. White, J. A., J. W. Schmidt and G. K. Bennett. Analysis of Queueing Systems. New York: Academic Press, 1975.

VIII. ACKNOWLEDGMENTS

I wish to express my gratitude to Dr. Eric M. Malstrom for providing and guiding this research. With his support and encouragement, the prolonged research for five years has been so interesting that the time is but a span. I also would like to express my appreciation to Dr. Way Kuo for his invaluable assistance both in academic and in private. My thanks are extended to the members of my Program of Study Committee, Dr. William H. Brockman, Dr. John C. Even, Jr., Dr. Howard D. Meeks, Dr. Terry A. Smay and Dr. Stephen B. Vardeman for their helpful advise. My doctoral study was partially supported by the Firestone Tire & Rubber Company (Des Moines Plant) through a research project for which Dr. Kuo was the principal investigator.

I would like to dedicate this dissertation to my parents for their love and continuous support throughout my graduate study. My study in the U.S. has been a wonderful experience. The harvest has been bountiful; not only academically but also spiritually. Last of all, I extend my deepest gratitude to my sisters, Shing-Dan and Shing-Fan, and brother, Du-Yan. With their spiritual support, although the flesh is ten thousand miles away from home, the mind is closer to one another than ever.

IX. APPENDIX A. PROGRAM LISTING FOR MIXED 0-1 INTEGER PROGRAMMING

This program for mixed 0-1 integer programming is designed to solve general linear programming problems of which continuous and binary variables are involved. This program uses two-phase Simplex method to solve for continuous solution, and Kochenberger's branch-and-bound procedure [63] is then carried out to obtain the binary integer solution.

Definition of program variables:

MO = maximum number of constraints
NO = maximum number of variables plus the number of > = constraints
NOS = maximum number of subproblems
In this program MO, NO and NOS are set to 40, 50 and 45, respectively. To increase the problem size, the following arrays must be adjusted.
M$(MO), N$(NO), A(MO), B(MO,NO), C(NO), P(NO), R(MO), T(NO), U(MO), X(NO), Z(NO), OPTV(NO), FV(NOS), CSTAR(NO), DT(NOS,NO)
N$(j) = the name of variable j assigned by the user
M$(i) = the name of constraint i
C(j) = coefficient of variable j in the objective function
B(i,j) = coefficient of variable j in constraint i
R(i) = right-hand-side value of constraint i
X(j) = shadow price in phase I; X(j) in phase II represents the solution value of variable j
Z(j) = shadow price of variable j in phase II
T(j) = coded names of nonbasic variables
A(i) = coded names of basic variables
DT(i,j) = δ value of binary variable j in subproblem i; δ = -1, 0, or 1.
CSTAR(j) = the new coefficient of binary variable j in the objective function
FV(i) = function value of subproblem i
U(i) = dual values
OPTV(j) = current best solution value of variable j
MAXZ = current best function value
PC = counter for cumulative number of subproblems
BN = branching node number
BVAR = branching variable number
BNY = number of binary variables

```
10  '*************************************************************************
20  '                                                                        '
30  '     Purpose - mixed 0-1 integer programming                            '
40  '                                                                        '
50  '     Algorithm - Kochenberger's branch-and-bound                        '
60  '                 procedure                                              '
70  '                                                                        '
80  '     Parameters -                                                       '
90  '                                                                        '
100 '             MO : number of constraints                                 '
110 '             NO : number of variables plus the number of                '
120 '                  >= constraints                                        '
130 '            NOS : number of subproblems                                 '
140 '           MAXZ : initial lower bound of the obj. fn.                   '
150 '                                                                        '
160 '     Input -                                                            '
170 '                                                                        '
180 '           > problem title                                              '
190 '           > MAX or MIN                                                 '
200 '           > number of variables                                        '
210 '           > obj. fn. coef., variable name                              '
220 '             (input binary vars. before continuous vars.)               '
230 '           > number of constraints                                      '
240 '           > name of constraint, type (>=<), RHS value                  '
250 '             (RHS values should >= 0)                                   '
260 '           > number of nonzero elements on LHS of all                   '
270 '             constraints                                                '
280 '           > nonzero coef., constraint name, var. name                  '
290 '                                                                        '
300 '*************************************************************************
310 '
320 DEFINT D
330 DIM M$(40),N$(50),A(40),B(40,50),C(50)
340 DIM P(50),R(40),T(50),U(40),X(50),Z(50)
350 DIM OPTV(50),FV(45),CSTAR(50),DT(45,50)
360 '
370 CLOSE #1 : OPEN "MIX.DAT" FOR INPUT AS #1
380 CLOSE #2 : OPEN "SOLUTION.DAT" FOR OUTPUT AS #2
390 INPUT "NUMBER OF BINARY VARIABLES: ",BNY
400 '
410 '          set up parameters
420 '
430 MO=40 : NO=50 : V7=0 : NOS=45
440 SMALL=.00001 : LARGE=.9999
450 MAXZ=-1000 : BIGM=1000
460 PC=1 : BN=1
470 FOR I=1 TO NO: OPTV(I)=0 : NEXT I
480 FOR I=1 TO NOS : FV(I)=0
490 FOR J=1 TO BNY : DT(I,J)=0
500 NEXT J : NEXT I
```

```
510 PRINT #2,"START TIME: ";DATE$,TIME$
520 '
530 '          read input data
540 '
550 PRINT "READING DATA..."
560 INPUT#1,T$ : PRINT #2,"TITLE ";T$ : ' read title
570 INPUT#1,Q$ : ' read MAX or MIN
580 INPUT#1,N : ' read number of variable
590 FOR J=1 TO N : INPUT#1,C(J),N$(J) : NEXT J
600 INPUT#1,M : ' read number of constraints
610 N3=0
620 FOR I=1 TO M
630 INPUT#1,M$(I),C$,R(I) : U(I)=0
640 IF C$="<=" THEN U(I)=1
650 IF C$="=" THEN U(I)=2
660 IF C$=">=" THEN U(I)=3
670 IF U(I) > 0 THEN 700
680 PRINT "*** ERROR IN THE ABOVE TYPE OF CONSTRAINT ***"
690 V7=V7+1 : GOTO 710
700 IF U(I)=3 THEN N3=N3+1
710 NEXT I
720 IF N+N3 <=NO THEN 750
730 PRINT "NUMBER OF VARIABLES EXCEEDS ",NO
740 V7=V7+1 : GOTO 820
750 FOR J=1 TO N : T(J)=J : NEXT J
760 L=0
770 FOR I=1 TO M
780 FOR J=1 TO N+N3 : B(I,J)=0 : NEXT J
790 IF U(I) <> 3 THEN 810
800 L=L+1 : B(I,N+L)=-1 : C(N+L)=0 : N$(N+L)=M$(I) : T(N+L)=N+I
810 NEXT I
820 INPUT#1,K1 : ' read total number of nonzero LHS coefs.
830 FOR L=1 TO K1
840 INPUT#1,V0,C$,D$
850 FOR I=1 TO M
860 IF C$=M$(I) THEN 900
870 NEXT I
880 V7=V7+1
890 PRINT C$;" ERROR, NOT CONSISTENT"
900 FOR J=1 TO N
910 IF D$=N$(J) THEN 940
920 NEXT J
930 V7=V7+1 : PRINT D$;" ERROR, NOT CONSISTENT" : GOTO 950
940 IF V7=0 THEN B(I,J)=V0
950 NEXT L
960 IF V7=0 THEN 1020
970 PRINT : PRINT V7;" ERRORS DETECTED, EXECUTION TERMINATED"
980 STOP
990 '
1000 '          set vector Z(j) to contain coefs. of objective fn.
```

```
1010 '
1020 N=N+N3 : FOR J=1 TO N : Z(J)=C(J)
1030 IF Q$="MIN" THEN Z(J)=-C(J)
1040 NEXT J
1050 FOR J=1 TO N
1060 X(J)=0
1070 FOR I=1 TO M
1080 IF U(I)=1 THEN 1100
1090 X(J)=X(J)-B(I,J)
1100 NEXT I
1110 NEXT J
1120 N9=0
1130 FOR I=1 TO M
1140 A(I)=-I
1150 IF U(I)=1 THEN 1170
1160 N9=N9+1 : A(I)=-I-M
1170 U(I)=0
1180 NEXT I
1190 FOR I=1 TO N : CSTAR(I)=C(I) : NEXT I
1200 '
1210 '          call LP subroutine
1220 '
1230 GOSUB 2080
1240 '
1250 '          start branch-and-bound procedure
1260 '
1270 CO=0 : FOR I=1 TO M : J=A(I)
1280 IF J <= 0 THEN 1300
1290 CO=CO+C(J)*R(I)
1300 NEXT I
1310 FV(BN)=CO : ' LP function value
1320 FOR J=1 TO N : X(J)=0 : NEXT J
1330 '
1340 '          LP continuous solution
1350 '
1360 FOR I=1 TO M
1370 K=A(I)
1380 IF K>0 AND K<=N-N3 THEN X(K)=R(I)
1390 NEXT I
1395 GOTO 1650
1400 '
1410 IF PC=NOS THEN 2030 : ' PC is the counter for subproblems
1420 IF PC=0 THEN 2040 : ' list is empty, deliver solution
1430 '
1440 '          select branching node
1450 '          (the node with highest bound for max. problems)
1460 '
1470 BN=1 : ' BN is the branching node
1480 IF PC=1 THEN 1540
1490 BN=1 : TMAX=FV(1)
```

```
1500 FOR I=2 TO PC
1510 IF TMAX >= FV(I) THEN 1530
1520 TMAX=FV(I) : BN=I
1530 NEXT I
1540 FOR J=1 TO BNY
1550 CSTAR(J)=DT(BN,J)*BIGM+C(J) : ' compute new obj. fn. coef.
1560 NEXT J
1570 FOR J=1 TO N
1580 IF T(J) <=0 THEN P(J)=-0 ELSE P(J)=-CSTAR(T(J))
1590 '
1600 '          perform sensitivity analysis for new obj. fn. coefs.
1610 '
1620 FOR I=1 TO M
1630 IF A(I)>0 AND A(I)<=N-N3 THEN P(J)=P(J)+CSTAR(A(I))*B(I,J)
1640 NEXT I : NEXT J : INDEX=2 : GOTO 2160
1650 IF FV(BN) <= MAXZ THEN 1800
1660 IF BNY=0 GOTO 1730
1670 FOR J=1 TO BNY
1680 IF (X(J) > SMALL AND X(J) < LARGE) AND DT(BN,J) <> 0 THEN 1800
1690 NEXT J
1700 FOR J=1 TO BNY
1710 IF X(J) > SMALL AND X(J) < LARGE THEN 1860
1720 NEXT J
1730 IF FV(BN) <= MAXZ THEN 1800
1740 MAXZ=FV(BN) : ' update the lower bound of the obj. fn.
1750 FOR I=1 TO N : OPTV(I)=X(I) : NEXT I
1760 IF NBY=0 GOTO 1810
1770 '
1780 '          fathom (eliminate the subproblem from the list)
1790 '
1800 FOR J=1 TO BNY : DT(BN,J)=DT(PC,J) : NEXT J
1805 FV(BN)=FV(PC)
1810 PC=PC-1 : GOTO 1410
1820 '
1830 '          select branching variable
1840 '          (most fractional integer variable)
1850 '
1860 FOR J=1 TO BNY
1870 IF X(J) > 1-X(J) THEN X(J)=1-X(J)
1880 NEXT J
1885 BVAR=1 : TMAX=X(1)
1890 IF BNY=1 THEN 1970
1900 FOR J=2 TO BNY
1910 IF TMAX > X(J) THEN 1930
1920 TMAX=X(J) : BVAR=J
1930 NEXT J
1940 '
1950 '          partition (add new subproblems to the list)
1960 '
1970 PC=PC+1
```

```
1980 FOR J=1 TO BNY : DT(PC,J)=DT(BN,J) : NEXT J
1990 DT(BN,BVAR)=1 : DT(PC,BVAR)=-1 : FV(PC)=FV(BN) : GOTO 1410
2000 '
2010 '          deliver the optimal solution
2020 '
2030 PRINT#2,"MAX. NUMBER OF SUBPROBLEMS ENCOUNTERED"
2040 PRINT#2,"FUNCTION VALUE: ";MAXZ : PRINT#2,"VARIABLE","VALUE"
2050 FOR I=1 TO N-N3 : PRINT#2,N$(I),OPTV(I) : NEXT I
2060 PRINT #2,"TERMINATING TIME: ";DATE$,TIME$
2070 END
2080 '
2090 '         Simplex algorithm (two-phase method)
2100 '
2110 I0=1 : INDEX=1
2120 FOR J=1 TO N : P(J)=-Z(J)
2130 FOR I=1 TO M : P(J)=P(J)+U(I)*B(I,J)
2140 NEXT I : NEXT J
2150 '
2160 '         pivot column
2170 '
2180 E9=-.0000001 : K9=0
2190 IF I0=2 THEN 2210
2200 IF N9 <= 0 THEN 3100
2210 FOR J=1 TO N
2220 IF T(J) < -M THEN 2310
2230 IF I0=2 THEN 2280
2240 IF X(J) >=E9 THEN 2310
2250 K9=J
2260 E9=X(J)
2270 GOTO 2310
2280 IF P(J) >=E9 THEN 2310
2290 K9=J
2300 E9=P(J)
2310 NEXT J
2320 IF K9 <= 0 THEN 3080
2330 '
2340 '         pivot row
2350 '
2360 K8=0
2370 C9=E9
2380 E9=1E+20
2390 FOR I=1 TO M
2400 IF B(I,K9) <= 0 THEN 2450
2410 R9=R(I)/B(I,K9)
2420 IF R9 >= E9 THEN 2450
2430 E9=R9
2440 K8=I
2450 NEXT I
2460 IF K8 > 0 THEN 2530
2470 PRINT
```

```
2480 PRINT#2,"*** SOLUTION UNBOUNDED ***"
2490 PRINT#2,"                         "
2500 STOP
2520 '
2530 '         transform tableau
2540 '
2550 V9=B(K8,K9)
2560 FOR J=1 TO N
2570 B(K8,J)=B(K8,J)/V9
2580 NEXT J
2590 R(K8)=R(K8)/V9
2600 FOR I=1 TO M
2610 IF I=K8 THEN 2670
2620 R(I)=R(I)-R(K8)*B(I,K9)
2630 FOR J=1 TO N
2640 IF J=K9 THEN 2660
2650 B(I,J)=B(I,J)-B(K8,J)*B(I,K9)
2660 NEXT J
2670 NEXT I
2680 FOR I=1 TO M
2690 B(I,K9)=-B(I,K9)/V9
2700 B(K8,K9)=1/V9
2710 NEXT I
2730 '
2740 '         interchange basic and nonbasic variables
2750 '
2760 R8=T(K9)
2770 T(K9)=A(K8)
2780 A(K8)=R8
2790 E9=Z(K9)
2800 Z(K9)=U(K8)
2810 U(K8)=E9
2820 IF T(K9) >= -M THEN 2840
2830 N9=N9-1
2840 IF IO=2 THEN 2930
2850 S9=P(K9)
2860 FOR J=1 TO N
2870 P(J)=P(J)-S9*B(K8,J)
2880 X(J)=X(J)-C9*B(K8,J)
2890 NEXT J
2900 P(K9)=-S9/V9
2910 X(K9)=-C9/V9
2920 GOTO 3000
2930 FOR J=1 TO N
2940 P(J)=P(J)-C9*B(K8,J)
2950 NEXT J
2960 P(K9)=-C9/V9
2970 '
2980 '         set very small elements of B(i,j) to zeros
2990 '
```

```
3000 FOR I=1 TO M
3010 FOR J=1 TO N
3020 IF ABS(B(I,J)) > .0000001 THEN 3040
3030 B(I,J)=0
3040 NEXT J
3050 NEXT I
3060 GOTO 2180
3070 '
3080 IF IO=2 AND INDEX=2 THEN 1270
3090 IF IO=2 THEN 3160
3100 IO=2
3110 IF N9 <= 0 THEN 2120
3120 PRINT
3130 PRINT#2," *** SOLUTION INFEASIBLE ***"
3140 PRINT#2,"     ------------------    "
3150 STOP
3160 RETURN
```

X. APPENDIX B. PROGRAM LISTING

FOR THE HEURISTIC DYNAMIC PROGRAMMING

This program employs the heuristic dynamic programming developed in this research to solve for a three-dimensional pallet problem. Two goals are involved in this program. The first goal is to maximize the utilization of a pallet cube. The second goal is to make the number of boxes of each type satisfy some user-specified number. If statements 2140-2220 are not in effect, goal 1 has higher priority than goal 2. Otherwise, goal 2 is more important than goal 1.

Definition of program variables:

LB(i),WB(i),HB(i)	= length, width and height of box type i, respectively
AREA(i)	= volume of box type i
F1(i,j,k)	= previous maximum return function value; i,j,k are the length, width and height of an index pallet, respectively. (i,j,k) varies from (1,1,1) to (L,W,H), the dimensions of the pallet.
F2(i,j,k)	= current maximum return function value
AV1(i,j,k,n)	= previous allocation vector of box type n
AV2(i,j,k,n)	= current allocation vector of box type n
UNIT(i,j)	= the unit vector
BA(i)	= volume occupied by boxes for combination i (see Figure 3.11)
RBOX(i,j)	= allocation vector (box type j) for combination i
RT(i)	= box ratio of combination i
NBOX(i)	= desired number of boxes of type i
TOTAL1(i)	= total number of boxes allocated for combination i
WA(i)	= sorted return function values in nonincreasing order
AL(i)	= allocation vector after sorting
INDEX(i)	= index number of sorted return function value; which stores the combination number that has i^{th} largest function value
MAX(i)	= current maximum return function value. If some allocation vector satisfies the selection criteria (Rules 1 to 4), then use MAX(1). Otherwise, use MAX(2).
NCA(i,j)	= current best solution of the allocation vector

```
10  '*****************************************************************************
20  '                                                                            '
30  '      Purpose - dynamic programming procedure for three                     '
40  '                dimensional pallet loading problem                          '
50  '                                                                            '
60  '      Parameters-                                                           '
70  '                                                                            '
80  '        F1(i,j,k) : previous maximum return function                        '
90  '                                                                            '
100 '        F2(i,j,k) : current maximum return function                         '
110 '                                                                            '
120 '                    i,j,k are length, width and height                      '
130 '                                                                            '
140 '        AV1(i,j,k,n) : previous allocation vector                           '
150 '                                                                            '
160 '        AV2(i,j,k,n) : current allocation vector                            '
170 '                                                                            '
180 '                       n is the the type number of box                      '
190 '                                                                            '
200 '      Revised date- April 22, 1987                                          '
210 '                                                                            '
220 '*****************************************************************************
230 '
240 DEFINT A-N,P,Q,S-Y
250 DIM LB(6),WB(6),HB(6),AREA(6),BA(7),RBOX(7,6),WA(7),UNIT(6,6)
260 DIM F1(4,4,4),F2(4,4,4),AV1(4,4,4,6),AV2(4,4,4,6),TOTAL1(7)
270 DIM INDEX(7),RT(7),NBOX(6),AL(6),MAX(2),NCA(2,6),TYPE(6)
280 '
290 '          input dimensions of pallet and boxes
300 '
310 INPUT "ENTER NUMBER OF VARIOUS BOX SIZES: ",NOB
320 J=1 : K=1 : BT=0 : ZOL=0
330 INPUT "PALLET DIMENSIONS: (LENGTH,WIDTH,HEIGHT): ",LP,WP,HP
340 IF K > NOB THEN GOTO 460
350 PRINT : PRINT ">>BOX TYPE: ";K
360 INPUT "BOX DIMENSIONS (LENGTH,WIDTH,HEIGHT): ",LB(J),WB(J),HB(J)
370 INPUT "DESIRED NUMBER ",NBOX(J)
380 TYPE(J)=K : BT=BT+1 : TOTAL2=TOTAL2+NBOX(J)
390 IF LB(J) <> WB(J) GOTO 410
400 J=J+1 : K=K+1 : GOTO 340
410 BT=BT+1 : NBOX(J+1)=NBOX(J) : LB(J+1)=WB(J) : WB(J+1)=LB(J)
420 HB(J+1)=HB(J) : TYPE(J+1)=K : J=J+2 : K=K+1 : GOTO 340
430 '
440 '          initialize
450 '
460 NBT=BT
470 FOR I3=0 TO HP : FOR I2=0 TO WP : FOR I1=0 TO LP
480 F1(I1,I2,I3)=0 : F2(I1,I2,I3)=0
490 FOR I4=1 TO NBT: AV1(I1,I2,I3,I4)=0: AV2(I1,I2,I3,I4)=0: NEXT I4
500 NEXT I1: NEXT I2: NEXT I3
```

```
510 FOR I2=1 TO NBT: FOR I1=1 TO NBT
520 UNIT(I1,I2)=0
530 NEXT I1 : NEXT I2
540 FOR I3=1 TO NBT: UNIT(I3,I3)=1: NEXT I3
550 FOR I3=1 TO NBT: AREA(I3)=LB(I3)*WB(I3)*HB(I3) : NEXT I3
560 '
570 '          determine function values and allocation vectors
580 '          of the first box type
590 '
600 FOR I3=1 TO HB(1): FOR I2=1 TO WB(1): FOR I1=1 TO LB(1)
610 F1(I1,I2,I3)=0 : AV1(I1,I2,I3,1)=0
620 NEXT I1 : NEXT I2 : NEXT I3
630 FOR I3=HB(1) TO HP : NHB=INT(I3/HB(1))
640 FOR I2=WB(1) TO WP : NWB=INT(I2/WB(1))
650 FOR I1=LB(1) TO LP : NLB=INT(I1/LB(1))
660 F1(I1,I2,I3)=NLB*NWB*NHB*AREA(1)
670 AV1(I1,I2,I3,1)=NLB*NWB*NHB
680 NEXT I1 : NEXT I2 : NEXT I3
690 I=2
700 IF NBT > 1 GOTO 740
710 F2(LP,WP,HP)=F1(LP,WP,HP) : AV2(LP,WP,HP,1)=AV1(LP,WP,HP,1)
720 GOTO 2450
730 '
740 '          enumerate all stages
750 '
760 IF I > NBT GOTO 2450
770 '
780 '          calculate fn. value given the index cube (ILX,IWX,IHX)
790 '
800 FOR IHX=1 TO HP : FOR IWX=1 TO WP : FOR ILX=1 TO LP
810 IF ILX < LB(I) OR IWX < WB(I) OR IHX < HB(I) THEN GOTO 830
820 GOTO 860
830 F2(ILX,IWX,IHX)=F1(ILX,IWX,IHX) : FOR I1=1 TO NBT
840 AV2(ILX,IWX,IHX,I1)=AV1(ILX,IWX,IHX,I1) : NEXT I1
850 GOTO 2360
860 '
870 '          allocate a box at coordinate (JPL,JPW,JPH)
880 '
890 I1=ILX-LB(I): I2=IWX-WB(I): I3=IHX-HB(I)
900 ICOL=INT(ILX/2) : ICOW=INT(IWX/2) : ICOH=INT(IHX/2)
910 IF I1 < ICOL THEN ICOL=I1
920 IF I2 < ICOW THEN ICOW=I2
930 IF I3 < ICOH THEN ICOH=I3
940 MAX(1)=0 : MAX(2)=0 : KY=2 : OPTR=100
950 FOR JPH=0 TO ICOH : FOR JPW=0 TO ICOW : FOR JPL=0 TO ICOL
960 XL=LB(I) : XW=WB(I) : XH=HB(I)
970 SL=ILX-XL-JPL : SW=IWX-XW-JPW : SH=IHX-XH-JPH
980 IF SL < 0 THEN SL=0
990 IF SW < 0 THEN SW=0
1000 IF SH < 0 THEN SH=0
```

```
1010 '
1020 '          calculate the function values
1030 '
1040 '                fn. value of combination 1 (Figure 3.11a)
1050 B1=F2(ILX,IWX,JPH): B2=F2(ILX,IWX,SH)
1060 B3=F2(JPL,IWX,XH) : B4=F2(SL,IWX,XH)
1070 B5=F2(XL,JPW,XH) : B6=F2(XL,SW,XH)
1080 BA(1)=AREA(I)+B1+B2+B3+B4+B5+B6
1090 '                fn. value of combination 2 (Figure 3.11b)
1100 B3=F2(JPL,XW,XH) : B4=F2(SL,XW,XH)
1110 B5=F2(ILX,JPW,XH) : B6=F2(ILX,SW,XH)
1120 BA(2)=AREA(I)+B1+B2+B3+B4+B5+B6
1130 '                fn. value of combination 3 (Figure 3.11c)
1140 B1=F2(JPL,IWX,IHX): B2=F2(SL,IWX,IHX)
1150 B3=F2(XL,JPW,IHX) : B4=F2(XL,SW,IHX)
1160 B5=F2(XL,XW,SH) : B6=F2(XL,XW,JPH)
1170 BA(3)=AREA(I)+B1+B2+B3+B4+B5+B6
1180 '                fn. value of combination 4 (Figure 3.11d)
1190 B3=F2(XL,JPW,XH) : B4=F2(XL,SW,XH)
1200 B5=F2(XL,IWX,SH) : B6=F2(XL,IWX,JPH)
1210 BA(4)=AREA(I)+B1+B2+B3+B4+B5+B6
1220 '                fn. value of combination 5 (Figure 3.11e)
1230 B1=F2(ILX,JPW,IHX): B2=F2(ILX,SW,IHX)
1240 B3=F2(JPL,XW,IHX) : B4=F2(SL,XW,IHX)
1250 B5=F2(XL,XW,SH) : B6=F2(XL,XW,JPH)
1260 BA(5)=AREA(I)+B1+B2+B3+B4+B5+B6
1270 '                fn. value of combination 6 (Figure 3.11f)
1280 B3=F2(JPL,XW,XH): B4=F2(SL,XW,XH)
1290 B5=F2(ILX,XW,SH) : B6=F2(ILX,XW,JPH)
1300 BA(6)=AREA(I)+B1+B2+B3+B4+B5+B6
1310 '                the previous best function value
1320 BA(7)=F1(ILX,IWX,IHX)
1330 GOSUB 2560
1340 '
1350 '          calculate the allocation vectors
1360 '
1370 '                allocation vector of combination 1
1380 FOR BT=1 TO NBT
1390 B1=AV2(ILX,IWX,JPH,BT): B2=AV2(ILX,IWX,SH,BT)
1400 B3=AV2(JPL,IWX,XH,BT): B4=AV2(SL,IWX,XH,BT): B5=AV2(XL,JPW,XH,BT)
1410 B6=AV2(XL,SW,XH,BT) : RBOX(1,BT)=UNIT(I,BT)+B1+B2+B3+B4+B5+B6
1420 NEXT BT
1430 '                allocation vector of combination 2
1440 FOR BT=1 TO NBT
1450 B1=AV2(ILX,IWX,JPH,BT): B2=AV2(ILX,IWX,SH,BT)
1460 B3=AV2(JPL,XW,XH,BT): B4=AV2(SL,XW,XH,BT): B5=AV2(ILX,JPW,XH,BT)
1470 B6=AV2(ILX,SW,XH,BT): RBOX(2,BT)=UNIT(I,BT)+B1+B2+B3+B4+B5+B6
1480 NEXT BT
1490 '                allocation vector of combination 3
1500 FOR BT=1 TO NBT
```

```
1510 B1=AV2(JPL,IWX,IHX,BT) : B2=AV2(SL,IWX,IHX,BT)
1520 B3=AV2(XL,JPW,IHX,BT) : B4=AV2(XL,SW,IHX,BT) : B5=AV2(XL,XW,SH,BT)
1530 B6=AV2(XL,XW,JPH,BT) : RBOX(3,BT)=UNIT(1,BT)+B1+B2+B3+B4+B5+B6
1540 NEXT BT
1550 '              allocation vector of combination 4
1560 FOR BT=1 TO NBT
1570 B1=AV2(JPL,IWX,IHX,BT) : B2=AV2(SL,IWX,IHX,BT)
1580 B3=AV2(XL,JPW,XH,BT) : B4=AV2(XL,SW,XH,BT) : B5=AV2(XL,IWX,SH,BT)
1590 B6=AV2(XL,IWX,JPH,BT) : RBOX(4,BT)=UNIT(1,BT)+B1+B2+B3+B4+B5+B6
1600 NEXT BT
1610 '              allocation vector of combination 5
1620 FOR BT=1 TO NBT
1630 B1=AV2(ILX,JPW,IHX,BT) : B2=AV2(ILX,SW,IHX,BT)
1640 B3=AV2(JPL,XW,IHX,BT) : B4=AV2(SL,XW,IHX,BT) : B5=AV2(XL,XW,SH,BT)
1650 B6=AV2(XL,XW,JPH,BT) : RBOX(5,BT)=UNIT(1,BT)+B1+B2+B3+B4+B5+B6
1660 NEXT BT
1670 '              allocation vector of combination 6
1680 FOR BT=1 TO NBT
1690 B1=AV2(ILX,JPW,IHX,BT) : B2=AV2(ILX,SW,IHX,BT)
1700 B3=AV2(JPL,XW,XH,BT) : B4=AV2(SL,XW,XH,BT) : B5=AV2(ILX,XW,SH,BT)
1710 B6=AV2(ILX,XW,JPH,BT) : RBOX(6,BT)=UNIT(1,BT)+B1+B2+B3+B4+B5+B6
1720 NEXT BT
1730 '              the previous best allocation vector
1740 FOR BT=1 TO NBT : RBOX(7,BT)=AV1(ILX,IWX,IHX,BT) : NEXT BT
1750 '
1760 '         calculate box ratios
1770 '
1780 FOR YY =1 TO 7 : TOTAL1(YY)=0 : NEXT YY
1790 FOR YY=1 TO 7 : FOR BT=1 TO NBT
1800 TOTAL1(YY)=TOTAL1(YY)+RBOX(YY,BT)
1810 NEXT BT : NEXT YY
1820 FOR J1=1 TO 7 : RT(J1)=0 : NEXT J1
1830 FOR J1=1 TO 7 : FOR BT=1 TO NBT
1840 IF TYPE(NBT-1)=TYPE(NBT) OR I=NBT GOTO 1860
1850 RT(J1)=RT(J1)+RBOX(J1,BT)/NBOX(BT) : GOTO 1880
1860 IF TOTAL1(J1)=0 THEN RT(J1)=RT(J1)+NBOX(BT)/TOTAL2 : GOTO 1880
1870 RT(J1)=RT(J1)+ABS(RBOX(J1,BT)/TOTAL1(J1)-NBOX(BT)/TOTAL2)
1880 NEXT BT : NEXT J1
1890 PNT=1 : PT=1
1900 IF PNT <= 7  GOTO 2030
1910 '
1920 '         if KY=1, some allocation vector satisfies Rule 4
1930 '         if KY=2, apply Rule 3
1940 '
1950 IF MAX(2) > LARGE OR KY=1 GOTO  2300
1960 MAX(2)=LARGE
1970 FOR BT=1 TO NBT : NCA(2,BT)=AL(BT) : NEXT BT
1980 GOTO 2300
1990 '
2000 '         apply Rule 1 to select the function value
```

```
2010 '
2020 RX1=RT(INDEX(PT))
2030 FOR J1=PT TO 6
2040 IF WA(J1) <> WA(J1+1) GOTO 2070
2050 IF RX1 > RT(INDEX(J+1)) THEN RX1=RT(INDEX(J+1)) : PNT=J1+1
2060 NEXT J1
2070 LARGE=BA(INDEX(PNT)) : RATIO=RT(INDEX(PNT))
2080 FOR BT=1 TO NBT : AL(BT)=RBOX(INDEX(PNT),BT): NEXT BT
2090 '
2100 '         apply Rule 4 : if allocated number > desired number
2110 '            for some box type, try the next best fn. value
2120 '         (go to 2230 if Rule 4 is not applied)
2130 '
2140 J=1 : K=1
2150 IF K > NOB THEN 2230
2160 IF J=NBT THEN GOTO 2210
2170 IF TYPE(J)<>TYPE(J+1) GOTO 2210
2180 IF AL(J)+AL(J+1) > (1+ZOL)*NBOX(K) GOTO 2200
2190 J=J+2 : K=K+1 : GOTO 2150
2200 PNT=PNT+1 : PT=PNT+1 : GOTO 1900
2210 IF AL(J) > (1+ZOL)*NBOX(K) GOTO 2200
2220 J=J+1 : K=K+1 : GOTO 2150
2230 KY=1
2240 '
2250 '         apply Rule 2
2260 '
2270 IF MAX(1) > LARGE OR (MAX(1)=LARGE AND OPTR < RATIO) GOTO 2300
2280 OPTR=RATIO : MAX(1)=LARGE
2290 FOR BT=1 TO NBT : NCA(1,BT)=AL(BT) : NEXT BT
2300 NEXT JPL : NEXT JPW : NEXT JPH
2310 '
2320 '         assign the best function value and allocation vector
2330 '
2340 F2(ILX,IWX,IHX)=MAX(KY)
2350 FOR BT=1 TO NBT: AV2(ILX,IWX,IHX,BT)=NCA(KY,BT): NEXT BT
2360 NEXT ILX : NEXT IWX: NEXT IHX
2370 '
2380 FOR H=1 TO HP: FOR W=1 TO WP: FOR L=1 TO LP
2390 F1(L,W,H)=F2(L,W,H)
2400 FOR BT=1 TO NBT: AV1(L,W,H,BT)=AV2(L,W,H,BT): NEXT BT
2410 NEXT L: NEXT W: NEXT H
2420 I=I+1
2430 GOTO 760 : ' start a new stage
2440 '
2450 '         deliver solution
2460 '
2470 PRINT "FUNCTION VALUE: ",F2(LP,WP,HP)
2480 PRINT "BOX SIZE","NUMBER"
2490 FOR BT=1 TO NBT
2500 PRINT LB(BT);" x ";WB(BT);" x ";HB(BT),AV2(LP,WP,HP,BT)
```

```
2510 NEXT BT
2520 END
2530 '
2540 '        sort the function values in nonincreasing order
2550 '
2560 FOR J1=1 TO 7: WA(J1)=BA(J1) : INDEX(J1)=J1 : NEXT J1
2570 FOR J1=1 TO 6 : L1=J1: JJ=J1+1
2580 FOR J2=JJ TO 7: IF WA(L1) >= WA(J2) GOTO 2600
2590 L1=J2
2600 NEXT J2
2610 STORE=WA(J1): WA(J1)=WA(L1): WA(L1)=STORE
2620 STORE=INDEX(J1) : INDEX(J1)=INDEX(L1) : INDEX(L1)=STORE
2630 NEXT J1
2640 RETURN
```

XI. APPENDIX C.

ROBOT CONTROL PROGRAM

This program allows the Rhino XR-2 robot to perform the palletizing task automatically. The Rhino conveyor, the turntable and the vacuum pick-up are also controlled by this program. The box type number is generated by a random number generator. It is assumed that boxes conveyed to the robot's pick-up position always follow a fixed orientation. The largest dimension of a box's length and width must be parallel to the conveyor's moving direction when the box is placed on the conveyor.

Definition of program variables:

```
INDEX          = index for moving speed via keyboard manipulation
H              = length between the base joint and the shoulder joint
L              = length of Rhino XR-2 robot's shoulder and elbow
LL             = length of the gripper
CNYH           = height of the conveyor
TABLEH         = height of the turntable
STORAGEH       = height of the storage area
HZ             = reference height above any obstruction
SRCH           = node number selected from the chain for placing
                 a box on the pallet
BT             = size number of a box currently considered
IX(i),IY(i),IZ(i) = initial position of storage area i along the
                 x-, the y- and the z-axis, respectively
LX(i),LY(i),LZ(i) = extreme position of storage area i along the
                 x-, y- and z-axis, respectively
LH(i),WH(i),HT(i) = the length, width and height of box size i,
                 respectively
IP(i)          = hard home position in encoder holes
P1(i)          = the point above the pick-up position in encoder
                 holes
P2(i,j)        = the pick-up position of box size i in encoder holes
SA(i,j)        = current placement location for storage area i
PX(i),PX(i),PZ(i) = placement location on a pallet for node #i along
                 the x-, y- and z-axis, respectively
PO(i)          = orientation of node i.
                 Do nothing if PO(i) = 0;
                 twist the gripper 90° if PO(i) = 1.
RANGE(i,j)     = box proportion of size j for distribution #i
NOPRE(i,j)     = number of predecessors of node j for pallet i
INDEX(i,j)     = forward pointer of node j for pallet i
INVR(i,j)      = backward pointer of node j for pallet i
START(i,j)     = start record address of chain/size j for pallet i
```

T(i)	= node # of the tail of a network branch i (working vector later on)
B(i)	= node # of the head of a network branch i
PNT(i)	= start record address of node i in the network branch file
PALLET(i)	= current number of boxes remaining to complete a full pallet cube
NSEQ(i)	= vector for generating random size number of box
PROB(i)	= cumulative box proportion
RT(i)	= offset number of encoder holes for a specific motor
FC(i,j)	= detect for closure of microswitch i (i = 1,2,3 for base, shoulder and elbow, respectively)
SPP$(i,j)	= moving number of encoder holes with positive direction for keyboard manipulation
SPN$(i,j)	= moving number of encoder holes with negative direction for keyboard manipulation
MTR$(i)	= motor name specified by i
SIGN$(i)	= moving direction of the motor specified by i
C(i)	= working vector storing the encoder holes to be moved

```
10 '*************************************************************************
20 '                                                                        '
30 '               Palletizing control program                              '
40 '                                                                        '
50 '                          for                                           '
60 '                                                                        '
70 '                     Rhino XR-2 robot                                   '
80 '                                                                        '
90 '*************************************************************************
100 '
110 DEFINT I,N,Q
120 DIM IX(4),IY(4),IZ(4),LX(4),LY(4),LZ(4)
130 DIM LH(4),WH(4),HT(4),HH(4),GN$(3),RT(3)
140 DIM IP(5),P1(5),P2(4,5),PU(5),PD(5)
150 DIM SA(4,3),JK(10,5),NSEQ(200),AP(5)
160 DIM PALLET(4),NOPRE(5,66),INDEX(5,66)
170 DIM INVR(5,66),PROB(4),RANGE(20,4),FC(3,3)
180 DIM TYPE(66),T(66),B(66),PNT(66),START(5,5)
190 DIM PX(66),PY(66),PZ(66),PO(66)
200 DIM SPP$(6,2),SPN$(6,2),MTR$(4),SIGN$(4)
210 DIM C(5),HO(3),IDX(3),MO$(3),SO$(3),SP$(2,2)
220 '
230 '          set up parameters
240 '
250 SPP$(1,1)="F+1" : SPP$(1,2)="F+10"
260 SPN$(1,1)="F-1" : SPN$(1,2)="F-10"
270 SPP$(2,1)="E+1" : SPP$(2,2)="E+10"
280 SPN$(2,1)="E-1" : SPN$(2,2)="E-10"
290 SPP$(3,1)="D+1" : SPP$(3,2)="D+10"
300 SPN$(3,1)="D-1" : SPN$(3,2)="D-10"
310 SPP$(4,1)="G+1" : SPP$(4,2)="G+10"
320 SPN$(4,1)="G-1" : SPN$(4,2)="G-10"
330 SPP$(5,1)="A+1" : SPP$(5,2)="A+5"
340 SPN$(5,1)="A-1" : SPN$(5,2)="A-5"
350 SPP$(6,1)="H+1" : SPP$(6,2)="H+10"
360 SPN$(6,1)="H-1" : SPN$(6,2)="H-10"
370 MTR$(1)="F" : MTR$(2)="E" : MTR$(3)="D" : MTR$(4)="A"
380 Z0$="6" : Z1=6 : Z2=95-Z1
390 INDEX=1 : QK=0
400 H=10.8 : L=9 : LL=6.25
410 PI=3.14159 : C=180!/PI : P=-90/C : R=0
420 SF=2620*C/360 : SE=3144*C/360 : SD=SE
430 SC=4541.3*C/360 : SA=4.1667*C
440 GN$(1)="-"  :  GN$(2)="+"  :  GN$(3)="-"
450 RT(1)=109 : RT(2)=91 : RT(3)=74
460 FC(1,1)=8 : FC(1,2)=10 : FC(1,3)=12
470 FC(2,1)=4 : FC(2,2)=6 : FC(2,3)=12
480 FC(3,1)=2 : FC(3,2)=6 : FC(3,3)=10
490 PRINT #1,"I"  :  A$=INPUT$(1,#1)  :  SWTCH=ASC(A$)
500 IF SWTCH < 32 THEN SWTCH=31 ELSE SWTCH=95
```

```
510 FOR J=1 TO 4 : HH(J)=0 : NEXT J
520 HZ=12 : AF=1 : NN=1 : QO=30 : Q1=QO : PRCTG=0!
530 TABLEH=7! : CNYH=3.75 : STORAGEH=4!
540 '
550 '                    dimensions of boxes
560 '
570 WH(1)=1.2 : LH(1)=1.2 : HT(1)=1
580 WH(2)=2.2 : LH(2)=1.2 : HT(2)=1
590 WH(3)=2.2 : LH(3)=2.2 : HT(3)=2
600 WH(4)=3.2 : LH(4)=2.2 : HT(4)=1
610 '
620 '                    initial and extreme locations of storages
630 '
640 IX(1)=9.5 : IY(1)=6 : IZ(1)=STORAGEH+1
650 IX(2)=12.5 : IY(2)=6 : IZ(2)=STORAGEH+1
660 IX(3)=5 : IY(3)=10 : IZ(3)=STORAGEH+2
670 IX(4)=1.5 : IY(4)=10 : IZ(4)=STORAGEH+1
680 LX(1)=10.7 : LY(1)=10.8 : LZ(1)=STORAGEH+4
690 LX(2)=12.5 : LY(2)=10.8 : LZ(2)=STORAGEH+4
700 LX(3)=7.2 : LY(3)=14.4 : LZ(3)=STORAGEH+8
710 LX(4)=1.5 : LY(4)=14.4 : LZ(4)=STORAGEH+4
720 '
730 '                    pick-up positions of various box sizes
740 '
750 X=9 : Y=0 : Z=13.5 : GOSUB 4230 : ' home position
760 IP(1)=T1 : IP(2)=T2 : IP(3)=T3 : IP(4)=T4 : IP(5)=ANGLE
770 GOSUB 6070
780 X=12 : Y=-2 : Z=12 : GOSUB 4230 : ' pt. above pick-up pos.
790 P1(1)=T1 : P1(2)=T2 : P1(3)=T3 : P1(4)=T4 : P1(5)=ANGLE
800 X=13.5 : Y=-1.5 : Z=CNYH+1 : GOSUB 4230 : ' pos. of type 1
810 P2(1,1)=T1 : P2(1,2)=T2 : P2(1,3)=T3
820 P2(1,4)=T4 : P2(1,5)=ANGLE
830 X=14 : Y=-1.5 : Z=CNYH+1 : GOSUB 4230 : ' pos. of type 2
840 P2(2,1)=T1 : P2(2,2)=T2 : P2(2,3)=T3
850 P2(2,4)=T4 : P2(2,5)=ANGLE
860 X=14 : Y=-1 : Z=CNYH+2 : GOSUB 4230 : ' pos. of type 3
870 P2(3,1)=T1 : P2(3,2)=T2 : P2(3,3)=T3
880 P2(3,4)=T4 : P2(3,5)=ANGLE
890 X=14.5 : Y=-1 : Z=CNYH+1 : GOSUB 4230 : ' pos. of type 4
900 P2(4,1)=T1 : P2(4,2)=T2 : P2(4,3)=T3
910 P2(4,4)=T4 : P2(4,5)=ANGLE
920 FOR J=1 TO 4
930 SA(J,1)=IX(J) : SA(J,2)=IY(J) : SA(J,3)=IZ(J)
940 NEXT J
950 FOR J=1 TO 5 : C(J)=P1(J)-IP(J) : NEXT J
960 C(5)=ABS(C(5)) : GOSUB 4490
970 '
980 '                    box proportions
990 '
1000 RANGE(1,1)=0 : RANGE(1,2)=1/3 : RANGE(1,3)=1/3 : RANGE(1,4)=1/3
```

```
1010 RANGE(2,1)=0 : RANGE(2,2)=1/3 : RANGE(2,3)=2/3 : RANGE(2,4)=0
1020 RANGE(3,1)=1/3 : RANGE(3,2)=0 : RANGE(3,3)=1/3 : RANGE(3,4)=1/3
1030 RANGE(4,1)=0 : RANGE(4,2)=2/3 : RANGE(4,3)=1/3 : RANGE(4,4)=0
1040 RANGE(5,1)=1/3 : RANGE(5,2)=1/3 : RANGE(5,3)=1/3 : RANGE(5,4)=0
1050 RANGE(6,1)=0 : RANGE(6,2)=2/3 : RANGE(6,3)=0 : RANGE(6,4)=1/3
1060 RANGE(7,1)=1/3 : RANGE(7,2)=0 : RANGE(7,3)=2/3 : RANGE(7,4)=0
1070 RANGE(8,1)=1/3 : RANGE(8,2)=1/3 : RANGE(8,3)=0 :RANGE(8,4)=1/3
1080 RANGE(9,1)=0 : RANGE(9,2)=1/3 : RANGE(9,3)=0 : RANGE(9,4)=2/3
1090 RANGE(10,1)=0: RANGE(10,2)=0: RANGE(10,3)=1/3: RANGE(10,4)=2/3
1100 RANGE(11,1)=1/3: RANGE(11,2)=2/3: RANGE(11,3)=0: RANGE(11,4)=0
1110 RANGE(12,1)=0: RANGE(12,2)=0: RANGE(12,3)=0: RANGE(12,4)=1
1120 RANGE(13,1)=1/3: RANGE(13,2)=0: RANGE(13,3)=0: RANGE(13,4)=2/3
1130 RANGE(14,1)=0: RANGE(14,2)=0: RANGE(14,3)=1: RANGE(14,4)=0
1140 RANGE(15,1)=0: RANGE(15,2)=1: RANGE(15,3)=0: RANGE(15,4)=0
1150 RANGE(16,1)=0: RANGE(16,2)=0: RANGE(16,3)=2/3: RANGE(16,4)=1/3
1160 RANGE(17,1)=2/3: RANGE(17,2)=1/3: RANGE(17,3)=0: RANGE(17,4)=0
1170 RANGE(18,1)=1: RANGE(18,2)=0: RANGE(18,3)=0: RANGE(18,4)=0
1180 RANGE(19,1)=2/3: RANGE(19,2)=0: RANGE(19,3)=1/3: RANGE(19,4)=0
1190 RANGE(20,1)=2/3: RANGE(20,2)=0: RANGE(20,3)=0: RANGE(20,4)=1/3
1200 '
1210 PR=1 : F1$="PLT" : F3$=".DAT"
1220 DATE$="1-1-1986" : TIME$="00:00:00"
1230 INPUT "TOTAL NUMBER OF PALLETS: ";TNP
1240 INPUT "ENTER NUMBER OF CYCLES FOR SELF-ADJUSTMENT: ";CYC
1250 '
1260 '          RS-232C asynchronous communication
1270 '
1280 COMFIL$="COM1:9600,E,7,2,DS"
1290 CLOSE #1
1300 OPEN COMFIL$ AS #1
1310 '
1320 '          keyboard manipulation
1330 '
1340 CLS : LOCATE 10,1
1350 PRINT "MANUAL OPERATION"
1360 PRINT
1370 PRINT "     BASE    SHLDR    ELBOW    PITCH    ROLL    TABLE"
1380 PRINT "        1        2        3        4       5        6 "
1390 PRINT
1400 PRINT "          Q        W        E        R       T        Y "
1410 PRINT
1420 PRINT "     PRESS 0--SLOW SPEED"
1430 PRINT "           9--FAST SPEED"
1440 PRINT : PRINT "     PRESS X TO EXIT"
1450 X$=INKEY$
1460 IF X$="" GOTO 1450
1470 IF X$="9" THEN INDEX=2
1480 IF X$="0" THEN INDEX=1
1490 IF X$="X" GOTO 1750
1500 IF X$ <> "1" GOTO 1520
```

```
1510  PRINT #1,SPP$(1,INDEX) : GOTO 1450
1520 IF X$ <> "Q" GOTO 1540
1530  PRINT #1,SPN$(1,INDEX) : GOTO 1450
1540 IF X$ <> "2" GOTO 1560
1550  PRINT #1,SPP$(2,INDEX) : GOTO 1450
1560 IF X$ <> "W" GOTO 1580
1570  PRINT #1,SPN$(2,INDEX) : GOTO 1450
1580 IF X$ <> "3" GOTO 1600
1590  PRINT #1,SPP$(3,INDEX) : GOTO 1450
1600 IF X$ <> "E" GOTO 1620
1610  PRINT #1,SPN$(3,INDEX) : GOTO 1450
1620 IF X$ <> "4" GOTO 1640
1630  PRINT #1,SPP$(4,INDEX) : GOTO 1450
1640 IF X$ <> "R" GOTO 1660
1650  PRINT #1,SPN$(4,INDEX) : GOTO 1450
1660 IF X$ <> "5" GOTO 1680
1670  PRINT #1,SPP$(5,INDEX) : GOTO 1450
1680 IF X$ <> "T" GOTO 1700
1690  PRINT #1,SPN$(5,INDEX) : GOTO 1450
1700 IF X$ <> "6" GOTO 1720
1710  PRINT #1,SPP$(6,INDEX) : GOTO 1450
1720 IF X$ <> "Y" GOTO 1450
1730  PRINT #1,SPN$(6,INDEX) : GOTO 1450
1740 '
1750 CLS
1760 '
1770 '        read data of a pallet pattern
1780 '
1790 INPUT "ENTER PALLET PATTERN NUMBER: ",FILEN : GOSUB 5660
1800 OPEN PATTERN$ FOR INPUT AS #3
1810 FOR J=1 TO QO : NSEQ(J)=J : NEXT J
1820 ANGL=360/TNP
1830 INPUT #3,TYPE,NODE,ACT
1840 E1=0
1850 FOR J=1 TO TYPE
1860 E1=E1+RANGE(FILEN,J) : PROB(J)=E1*QO
1870 NEXT J
1880 FOR I=1 TO NODE
1890 INPUT #3,TE,TYPE(I),PX(I),PY(I),PZ(I),PO(I) : PNT(I)=0
1900 NEXT I
1910 FOR I=1 TO ACT : INPUT #3,T(I),B(I) : NEXT I
1920 FOR I=1 TO 5 : FOR J=1 TO 5 : START(I,J)=0 : NEXT J : NEXT I
1930 MAX=NODE : IF MAX < ACT THEN MAX=ACT
1940 CLOSE #3
1950 '
1960 '        construct chains
1970 '
1980 FOR I=1 TO TNP+1 : FOR J=1 TO MAX
1990 NOPRE(I,J)=0 : INDEX(I,J)=0 : INVR(I,J)=0
2000 NEXT J : NEXT I
```

```
2010 B(ACT+1)=0 : T(ACT+1)=0 : J=1 : NO=T(1)
2020 NOPRE(1,B(1))=NOPRE(1,B(1))+1
2030 FOR I=2 TO ACT+1
2040 NOPRE(1,B(I))=NOPRE(1,B(I))+1
2050 IF NO=T(I) GOTO 2070
2060 PNT(NO)=J : NO=T(I) : J=I
2070 NEXT I
2080 FOR I=1 TO TYPE : T(I)=0 : NEXT I
2090 FOR I=1 TO NODE
2100 IF T(TYPE(I))=0 THEN START(1,TYPE(I))=I
2110 INDEX(1,T(TYPE(I)))=I : T(TYPE(I))=I
2120 NEXT I
2130 FOR I=1 TO TYPE : T(I)=0 : NEXT I
2140 FOR I=NODE TO 1 STEP -1
2150 INVR(1,T(TYPE(I)))=I : T(TYPE(I))=I
2160 NEXT I
2165 INDEX(1,0)=0
2170 FOR I=2 TO TNP+1 : FOR J=0 TO MAX
2180 NOPRE(I,J)=NOPRE(1,J)
2190 INDEX(I,J)=INDEX(1,J)
2200 INVR(I,J)=INVR(1,J)
2210 NEXT J : NEXT I
2220 FOR I=2 TO TNP+1 : FOR J=1 TO TYPE
2230 START(I,J)=START(1,J)
2240 NEXT J : NEXT I
2250 FOR J=1 TO TNP : PALLET(I)=NODE : NEXT I
2260 '
2270 '         pick up a box from in-feeding conveyor
2280 '
2290 QK=QK+1
2300 IF QK MOD CYC =0 OR QK = 1 THEN GOSUB 5990
2310 GOSUB 5810 : ' generate a box type
2320 FOR J=1 TO 5 : C(J)=P2(BT,J)-P1(J) : NEXT J
2330 C(5)=ABS(C(5)) : GOSUB 4490
2340 PRINT #1,"C+09" : GOSUB 5760 : ' delay for picking up a box
2350 FOR J=1 TO 4 : C(J)=-C(J) : NEXT J : GOSUB 4490
2360 '
2370 '         determine where to place boxes
2380 '         (place the box onto pallet if NC=0;
2390 '         place the box in the storage area if NC=1)
2400 '
2410 GOSUB 3330
2420 IF NC=0 GOTO 3260
2430 IF NC=1 GOTO 2460
2440 PRINT "ERROR" : STOP
2450 '
2460 '         place a box in the storage area
2470 '
2480 X=SA(BT,1) : Y=SA(BT,2) : Z=HZ : GOSUB 4230
2490 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
```

```
2500 FOR J=1 TO 5 : C(J)=PU(J)-P1(J) : NEXT J
2510 C(5)=ABS(C(5)) : GOSUB 4490
2520 X=SA(BT,1) : Y=SA(BT,2) : Z=SA(BT,3)+AF : GOSUB 4230
2530 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
2540 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
2550 C(5)=ABS(C(5)) : GOSUB 4490
2560 FOR J=1 TO 5 : T(J)=PD(J) : NEXT J
2570 TX=SA(BT,1) : TY=SA(BT,2) : TZ=SA(BT,3)
2580 GOSUB 5160 : ' go straight down
2590 PRINT #1,"CX" : GOSUB 5760 : ' delay for placing a box
2600 GOSUB 5320 : ' go straight up
2610 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
2620 C(5)=ABS(C(5)) : GOSUB 4490
2630 '
2640 '                update next placement location in storages
2650 '
2660 SA(BT,1)=SA(BT,1)+WH(BT)
2670 IF SA(BT,1) <= LX(BT) GOTO 2840
2680 SA(BT,1)=IX(BT) : SA(BT,2)=SA(BT,2)+LH(BT)
2690 IF SA(BT,2) <= LY(BT) GOTO 2840
2700 SA(BT,2)=IY(BT) : SA(BT,3)=SA(BT,3)+HT(BT)
2710 IF SA(BT,3) <= LZ(BT) GOTO 2840
2720 '
2730 '                storage overflow
2740 '
2750 CLS : PRINT "STORAGE OVERFLOW"
2760 PRINT : PRINT "CLEAN THE STORAGE"
2770 FOR J=1 TO 3 : PLAY "C+L2" : NEXT J
2780 PRINT : PRINT "ENTER ANY KEY TO CONTINUE " : INPUT TE : CLS
2790 SA(BT,1)=IX(BT) : SA(BT,2)=IY(BT) : SA(BT,3)=IZ(BT)
2800 FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
2810 C(5)=ABS(C(5)) : GOSUB 4490
2820 GOTO 2270 : ' go back to pick-up position
2830 '
2840 '         remove one box from every storage area
2850 '         and place it on the pallet
2860 '
2870 PR1=0
2880 ICUM=0
2890 FOR N=1 TO TYPE
2900 IF RANGE(FILEN,N)=0 THEN ICUM=ICUM+1 : GOTO 3190
2910 IF SA(N,1)=IX(N) AND SA(N,2)=IY(N) AND SA(N,3)=IZ(N) THEN
                      ICUM=ICUM+1 : GOTO 3190
2930 BT=N : GOSUB 3330
2940 IF NC=1 THEN ICUM=ICUM+1 : GOTO 3190 : 'no pallet space avail.
2950 '
2960 '                det. location of the box to be removed
2970 '
2980 SA(N,1)=SA(N,1)-WH(N)
2990 IF SA(N,1) >= IX(N) GOTO 3050
```

```
3000 SA(N,1)=LX(N) : SA(N,2)=SA(N,2)-LH(N)
3010 IF SA(N,2) >= IY(N) GOTO 3050
3020 SA(N,2)=LY(N) : SA(N,3)=SA(N,3)-HT(N)
3030 IF SA(N,3) >= IZ(N) GOTO 3050
3040 SA(N,1)=IX(N) : SA(N,2)=IY(N) : SA(N,3)=IZ(N)
3050 X=SA(N,1) : Y=SA(N,2) : Z=HZ : GOSUB 4230
3060 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
3070 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
3080 C(5)=ABS(C(5)) : GOSUB 4490
3090 X=SA(N,1) : Y=SA(N,2) : Z=SA(N,3)+AF : GOSUB 4230
3100 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
3110 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
3120 C(5)=ABS(C(5)) : GOSUB 4490
3130 FOR J=1 TO 5 : T(J)=PU(J) : NEXT J
3140 TX=SA(N,1) : TY=SA(N,2) : TZ=SA(N,3) : GOSUB 5160
3150 PRINT #1,"C+09" : GOSUB 5760 : GOSUB 5320
3160 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
3170 C(5)=ABS(C(5)) : GOSUB 4490
3180 GOSUB 3690 : ' move to pallet
3190 NEXT N
3200 IF ICUM <> TYPE AND PALLET(PR)/NODE <= PRCTG THEN
                      PR1=1 : GOTO 2880
3220 FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
3230 C(5)=ABS(C(5)) : GOSUB 4490
3240 GOTO 2270 : ' go to pick-up position
3250 '
3260 '         the pallet
3270 '
3280 FOR J=1 TO 5 : PD(J)=P1(J) : NEXT J
3290 GOSUB 3690 : ' place boxes on pallet
3300 GOTO 2840 : ' move one box from every storage
3310 END
3320 '
3330 '         search the chain to det. box placement location
3340 '
3350 T(1)=PR : TE=PR : NC=0
3360 FOR I=2 TO TNP
3370 TE=TE+1 : IF TE > TNP THEN TE=1
3380 T(I)=TE : NEXT I
3390 FOR K=1 TO TNP : TE=T(K) : SEQ=K
3400 SRCH=START(TE,BT)
3410 FOR I=1 TO NODE
3420 IF START(TE,BT)=0 GOTO 3480
3430 IF NOPRE(TE,SRCH)=0 THEN GOTO 3520
3440 SRCH=INDEX(TE,SRCH)
3450 IF SRCH=0 GOTO 3480
3460 NEXT I
3470 IF PR1=1 GOTO 3490
3480 NEXT K
3490 NC=1
```

```
3500 RETURN : ' no pallet space available
3510 '
3520 '                  update the chain
3530 '
3540 IF START(TE,BT)=SRCH THEN
                     START(TE,BT)=INDEX(TE,SRCH) : GOTO 3580
3560 INDEX(TE,INVR(TE,SRCH))=INDEX(TE,SRCH)
3570 INVR(TE,INDEX(TE,SRCH))=INVR(TE,SRCH)
3580 IF PNT(SRCH)=0 GOTO 3660
3585 NXT=SRCH
3590 FOR I=1 TO NODE
3595 NXT=NXT+1
3600 IF PNT(NXT) <> 0 THEN NO=PNT(NXT)-1 : GOTO 3630
3610 IF NXT=NODE+1 THEN NO=ACT : GOTO 3630
3620 NEXT I : PRINT "ERROR ON PNT" : STOP
3630 FOR K=PNT(SRCH) TO NO
3640 NOPRE(TE,B(K))=NOPRE(TE,B(K))-1
3650 NEXT K
3660 PLTN=TE : NC=0
3670 RETURN
3680 '
3690 '         place the box onto the pallet
3700 '         rotate the turntable if necessary
3710 '
3720 IF ANGL*(SEQ-1) <= 180 GOTO 3740
3730 SP$(1,1)="-" : RTH=(360/ANGL-SEQ+1)*2620/TNP : GOTO 3750
3740 SP$(1,1)="+" : RTH=2620/TNP*(SEQ-1)
3750 GOSUB 5400 : ' rotate table for desired pallet
3760 X=PX(SRCH) : Y=PY(SRCH) : Z=HZ : GOSUB 4230
3770 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
3780 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
3790 C(5)=ABS(C(5)) : GOSUB 4490
3800 IF PO(SRCH) <> 1 GOTO 3820
3810 SP$(2,2)="+" : GOSUB 5530
3820 X=PX(SRCH) : Y=PY(SRCH) : Z=PZ(SRCH)+TABLEH+AF : GOSUB 4230
3830 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
3840 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
3850 C(5)=ABS(C(5)) : GOSUB 4490
3860 FOR J=1 TO 5 : T(J)=PD(J) : NEXT J
3870 TX=PX(SRCH) : TY=PY(SRCH) : TZ=PZ(SRCH)+TABLEH : GOSUB 5160
3880 PRINT #1,"CX" : GOSUB 5760 : GOSUB 5320
3890 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
3900 C(5)=ABS(C(5)) : GOSUB 4490
3910 IF SP$(1,1)="+" THEN SP$(1,1)="-" ELSE SP$(1,1)="+"
3920 GOSUB 5400 : ' rotate table back to the original pallet
3930 PALLET(PLTN)=PALLET(PLTN)-1
3940 IF PO(SRCH) <> 1 GOTO 3960
3950 SP$(2,2)="-" : GOSUB 5530
3960 IF PALLET(PLTN) <> 0 THEN RETURN
3970 '
```

```
3980 '         pallet is full
3990 '
4000 FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
4010 C(5)=ABS(C(5)) : GOSUB 4490
4020 CLS : LOCATE 10,5 : PRINT "PALLET IS FULL"
4030 FOR J=1 TO 3 : PLAY "A+L4" : NEXT J
4040 IF PR=TNP THEN PR=1 ELSE PR=PR+1 : ' update pallet priority
4050 NO=TNP+1
4060 '
4070 '                 restore the parameters of the chain
4080 '
4090 FOR J=0 TO MAX
4100 NOPRE(PLTN,J)=NOPRE(NO,J)
4110 INDEX(PLTN,J)=INDEX(NO,J)
4120 INVR(PLTN,J)=INVR(NO,J)
4130 NEXT J
4140 FOR J=1 TO TYPE : START(PLTN,J)=START(NO,J) : NEXT J
4150 PALLET(PLTN)=NODE
4160 IF TNP <> 1 THEN RTH=2620/TNP ELSE RTH=655
4170 SP$(1,1)="+" : GOSUB 5400 : ' rotate the turntable
4180 PRINT "REMOVE THE PALLET AND INSERT A NEW PALLET" : PRINT
4190 PRINT "ENTER ANY KEY WHEN READY" : INPUT TE : CLS
4200 GOTO 2270
4210 RETURN
4220 '
4230 '          coordinate transformation
4240 '
4250 RR=SQR(X*X+Y*Y)
4260 TE=ABS(H-Z-LL)/181
4270 TE=-ATN(TE/SQR(-TE*TE+1))+1.5708
4280 TE=18*SIN(TE)
4290 IF RR > TE THEN CLS : PRINT "ERROR ON XYZ-COOR" : STOP
4300 IF X = 0 THEN T1=SGN(Y)*PI/2
4310 IF X > 0 THEN T1=ATN(Y/X)
4320 IF X < 0 AND Y > 0 THEN T1=PI-ATN(Y/ABS(X))
4330 IF X < 0 AND Y < 0 THEN T1=-(PI-ATN(Y/X))
4340 ANGLE=ABS(T1*C)
4345 IF X=0 THEN T4=0 ELSE T4=ATN(ABS(Y/X))
4350 RO=RR
4360 ZO=Z+LL-H
4370 IF RO=0 THEN G=SGN(ZO)*PI/21 ELSE G=ATN(ZO/RO)
4380 A=RO*RO+ZO*ZO
4390 A=4*L*L/A-1
4400 A=ATN(SQR(A))
4410 T2=A+G
4420 T3=G-A
4430 T1=INT(T1*SF)
4440 T2=INT(T2*SE)
4450 T3=INT(T3*SD)
4460 T4=INT(T4*SA)
```

```
4470 RETURN
4480 '
4490 '         simultaneous movements of base, shoulder and elbow
4500 '
4510 SIGN$(1)="+" : SIGN$(2)="+" : SIGN$(3)="+" : SIGN$(4)="+"
4520 IF C(1) < 0 THEN SIGN$(1)="-"
4530 IF C(2) > 0 THEN SIGN$(2)="-"
4540 IF C(3) < 0 THEN SIGN$(3)="-"
4550 IF C(5) <= 90 AND C(1) < 0 THEN SIGN$(4)="-"
4560 IF C(5) <= 90 AND C(1) > 0 THEN SIGN$(4)="+"
4570 IF C(5) > 90 AND C(1) < 0 THEN SIGN$(4)="+"
4580 IF C(5) > 90 AND C(1) > 0 THEN SIGN$(4)="-"
4590 '
4600 '                 sort encoder holes in nonincreasing order
4610 '
4620 HO(1)=ABS(C(1)) : HO(2)=ABS(C(2)) : HO(3)=ABS(C(3))
4630 IDX(1)=1 : IDX(2)=2 : IDX(3)=3
4640 FOR J1=1 TO 2 : L1=J1 : JJ=J1+1 : FOR J2=JJ TO 3
4650 IF HO(L1) < HO(J2) THEN L1=J2
4660 NEXT J2
4670 TE=HO(J1) : HO(J1)=HO(L1) : HO(L1)=TE
4680 TE=IDX(J1) : IDX(J1)=IDX(L1) : IDX(L1)=TE
4690 NEXT J1
4700 FOR I=1 TO 3
4710 MO$(I)=MTR$(IDX(I)) : SO$(I)=SIGN$(IDX(I))
4720 NEXT I
4730 '
4740 IF HO(1)=0 GOTO 4990
4750 E1=0 : COUNT=0 : RATE1=HO(2)/HO(1) : RATE2=HO(3)/HO(1)
4760 D1=0 : HS=HO(1)
4770 PRINT #1,MO$(1);"?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
4780 IF CD > Z2 GOTO 4770
4790 IF HS > Z1 GOTO 4820
4800 TN$=CHR$(HS+48) : PRINT #1,MO$(1);SO$(1);TN$
4810 COUNT=COUNT+HS : GOTO 4840
4820 PRINT #1,MO$(1);SO$(1);ZO$
4830 COUNT=COUNT+Z1
4840 HS=HS-Z1
4850 IF HO(2)=0 GOTO 4950
4860 E2=INT(COUNT*RATE1) : TE=E2-E1
4870 IF TE=0 GOTO 4900
4880 E1=E2 : TN$=CHR$(TE+48)
4890 PRINT #1,MO$(2);SO$(2);TN$
4900 IF HO(3)=0 GOTO 4950
4910 D2=INT(COUNT*RATE2) : TE=D2-D1
4920 IF TE=0 GOTO 4950
4930 D1=D2 : TN$=CHR$(TE+48)
4940 PRINT #1,MO$(3);SO$(3);TN$
4950 IF HS > 0 GOTO 4770
4960 '
```

```
4970 '                roll gripper to be parallel to the y-axis
4980 '
4990 HS=ABS(C(4)) : IF HS=0 GOTO 5090
5000 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5010 IF CD > Z2 GOTO 5000
5020 IF HS > Z1 GOTO 5060
5030 TN$=CHR$(HS+48)
5040 PRINT #1,MTR$(4);SIGN$(4);TN$
5050 GOTO 5090
5060 PRINT #1,MTR$(4);SIGN$(4);Z0$
5070 HS=HS-Z1
5080 IF HS > 0 GOTO 5000
5090 CD=0
5100 FOR I=1 TO 4 : PRINT #1,MTR$(I);"?"
5110 TN$=INPUT$(1,#1) : CD=CD+ASC(TN$)-32
5120 NEXT I
5130 IF CD <> 0 GOTO 5090
5140 RETURN
5150 '
5160 '         move the arm straight down
5170 '
5180 JM=AF/NN : SZ=TZ+AF
5190 FOR J=1 TO NN
5200 X=TX : Y=TY : Z=SZ-JM*J : GOSUB 4230
5210 TYPE(1)=T1 : TYPE(2)=T2 : TYPE(3)=T3
5220 TYPE(4)=T4 : TYPE(5)=ANGLE
5230 FOR JP=1 TO 5 : JK(J,JP)=TYPE(JP)-T(JP) : NEXT JP
5240 JK(J,5)=ABS(JK(J,5))
5250 FOR JP=1 TO 5 : T(JP)=TYPE(JP) : NEXT JP
5260 NEXT J
5270 FOR J=1 TO NN
5280 FOR JP=1 TO 5 : C(JP)=JK(J,JP) : NEXT JP : GOSUB 4490
5290 NEXT J
5300 RETURN
5310 '
5320 '         move the arm straight up
5330 '
5340 FOR J=NN TO 1 STEP -1
5350 FOR JP=1 TO 4 : C(JP)=-JK(J,JP) : NEXT JP
5360 C(5)=ABS(JK(J,5)) : GOSUB 4490
5370 NEXT J
5380 RETURN
5390 '
5400 '         rotate the turntable
5410 '
5420 HS=RTH
5430 PRINT #1,"H?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5440 IF CD > 85 GOTO 5430
5450 IF HS >= 10 GOTO 5470
5460 TN$=CHR$(HS+48) : PRINT #1,"H";SP$(1,1);TN$ : GOTO 5490
```

```
5470 PRINT #1,"H";SP$(1,1);"10" : HS=HS-10
5480 IF HS > 0 GOTO 5430
5490 PRINT #1,"H?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5500 IF CD <> 0 GOTO 5490
5510 RETURN
5520 '
5530 '        roll the gripper to change box orientation
5540 '
5550 HS=375
5560 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5570 IF CD > 85 GOTO 5560
5580 IF HS >= 6 GOTO 5600
5590 TN$=CHR$(HS+48) : PRINT #1,"A";SP$(2,2);TN$ : GOTO 5620
5600 PRINT #1,"A";SP$(2,2);"6" : HS=HS-6
5610 IF HS > 0 GOTO 5560
5620 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5630 IF CD <> 0 GOTO 5620
5640 RETURN
5650 '
5660 '        convert number to characters for
5670 '        determining file name of a pallet pattern
5690 '
5700 IF FILEN < 10 THEN F2$=CHR$(FILEN+48) : GOTO 5730
5710 E1=INT(FILEN/10) : R10$=CHR$(E1+48)
5720 E2=FILEN-E1*10 : R01$=CHR$(E2+48) : F2$=R10$+R01$
5730 PATTERN$=F1$+F2$+F3$
5740 RETURN
5750 '
5760 '        delay for picking up and placing a box
5770 '
5780 FOR J=1 TO 500 : E1=SQR(2) : NEXT J
5790 RETURN
5800 '
5810 '        generate box sizes using random number generator
5820 '
5830 E1=INT(RND*Q1)+1
5840 FOR J=1 TO TYPE
5850 IF NSEQ(E1)<=PROB(J) THEN BT=J : GOTO 5880
5860 NEXT J
5870 PRINT "ERROR ON BT"
5880 HH(BT)=HH(BT)+1
5890 FOR J=1 TO BT : PLAY "G+L6" : NEXT J
5900 CLS : E9=350 : E2=30
5910 FOR J=1 TO BT : CIRCLE (E9,E2),20,BT : PAINT (E9,E2),BT
5920 E2=E2+35 : NEXT J
5930 NSEQ(E1)=NSEQ(Q1) : Q1=Q1-1
5940 IF Q1 <> 0 THEN RETURN
5950 Q1=Q0 : CLS : PRINT "END OF THE DISTRIBUTION"
5960 INPUT "ENTER ANY KEY TO CONTINUE";TE
5970 GOTO 1790
```

```
5980 RETURN
5990 '
6000 FOR J=1 TO 5 : C(J)=IP(J)-P1(J) : NEXT J : ' back to home pos.
6010 C(5)=ABS(C(5)) : GOSUB 4490
6020 GOSUB 6070
6030 FOR J=1 TO 5 : C(J)=P1(J)-IP(J) : NEXT J
6040 C(5)=ABS(C(5)) : GOSUB 4490
6050 RETURN
6060 '
6070 '        self-adjust to hard home position
6080 '
6090 PRINT #1,"I" : A$=INPUT$(1,#1) : INTRP=ASC(A$)
6100 SWTCH=31 : IF INTRP > 31 THEN SWTCH=95
6110 FR=SWTCH-INTRP : G$="+"
6120 IF FR=14 OR FR=FC(1,1) THEN J=1:HS=120:GOSUB 6460
6130 IF FR=FC(1,2) OR FR=FC(1,3) THEN J=1 : HS=120 : GOSUB 6460
6140 IF FR=14 OR FR=FC(2,1) THEN J=2:HS=100:GOSUB 6460
6150 IF FR=FC(2,2) OR FR=FC(2,3) THEN J=2 : HS=100 : GOSUB 6460
6160 IF FR=14 OR FR=FC(3,1) THEN J=3:HS=85:GOSUB 6460
6170 IF FR=FC(3,2) OR FR=FC(3,3) THEN J=3 : HS=85 : GOSUB 6460
6180 '
6190 FOR I=1 TO 3 : J=I
6200 ADJ=150
6210 TE=0 : S$=GN$(I) : ID1=0
6220 PRINT #1,MTR$(I);S$;"2" : TE=TE+2
6230 PRINT #1,"I" : B$=INPUT$(1,#1) : INTRP=ASC(B$)
6240 SWTCH=31 : IF INTRP > 31 THEN SWTCH=95
6250 IF TE < ADJ GOTO 6340
6260 G$="+" : IF S$="+" THEN G$="-"
6270 HS=ADJ : GOSUB 6460 : ID1=ID1+1
6280 IF ID1=2 AND ADJ=150 THEN ADJ=250 : GOTO 6210
6290 IF ID1=2 AND ADJ>=250 THEN
            INPUT "ENTER NEW ADJ VALUE: ",ADJ : GOTO 6210
6310 TE=0
6320 IF S$="+" THEN S$="-" ELSE S$="+"
6330 GOTO 6220
6340 FR=SWTCH-INTRP
6350 IF FR=14 OR FR=FC(1,1) GOTO 6400
6370 IF FR=FC(1,2) OR FR=FC(1,3) GOTO 6400
6390 GOTO 6220
6400 IF S$=GN$(I) GOTO 6420
6410 HS=RT(I) : G$=S$ : GOSUB 6460
6420 NEXT I
6430 PRINT #1,"AX" : PRINT #1,"DX" : PRINT #1,"EX"
6440 PRINT #1,"FX" : PRINT #1,"GX" : PRINT #1,"HX"
6450 RETURN
6460 '
6470 PRINT #1,MTR$(J);"?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6480 IF CD > 70 GOTO 6470
6490 IF HS > 9 GOTO 6510
```

```
6500 TN$=CHR$(HS+48) : PRINT #1,MTR$(J);G$;TN$ : GOTO 6530
6510 PRINT #1,MTR$(J);G$;"10" : HS=HS-10
6520 IF HS > 0 GOTO 6470
6530 PRINT #1,MTR$(J);"?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6540 IF CD <> 0 GOTO 6530
6550 RETURN
6560 END
```

XII. APPENDIX D.

SIMULATION PROGRAM LISTING

(KNOWN BOX SIZE DISTRIBUTIONS)

This program is employed to simulate the robotic palletizing operations with various number of simultaneously loaded pallets (multi-pallet packing). The length and box proportions of a distribution run can be predetermined, and are part of the input to this program. The pallet pattern can only be changed when a new distribution run starts.

Definition of program variables:

TNP = total number of simultaneously loaded pallets
DNEWTM = current time
FILEN = current distribution/pallet pattern number used

The definition of remaining variables can be found in the program and Appendix C.

```
10 '*************************************************************************
20 '                                                                        '
30 '   Purpose - Robotic palletizing simulation program                     '
40 '             for multi-pallet packing with                              '
50 '             known box distributions                                    '
60 '                                                                        '
70 '   Parameters - The following variable names with suffix 1              '
80 '                are used to collect the statistics of a                 '
90 '                distribution run.  The variable names with              '
100 '               suffix 2 are used to collect the statistics             '
110 '               of total simulation.                                    '
120 '                                                                       '
130 '        NCLD : total number of boxes loaded to the pallet              '
140 '         STM : start time of the simulation                            '
150 '         CCT : cumulative cycle times                                  '
160 '        OPTS : cumulative operation times of directly                  '
170 '               loading boxes to the pallet                             '
180 '         CTM : cumulative waiting times                                '
190 '               (All statistics are continuous from the                 '
200 '               previous distribution)                                  '
210 '        DCTM : cumulative waiting times                                '
220 '               (independent of the previous distribution;              '
230 '               all statistics are set back to zero at the              '
240 '               beginning of a new distribution run)                    '
250 '        NGES : total boxes generated                                   '
260 '        NMAX : maximum contents (dependent stat.)                      '
270 '       DNMAX : maximum contents (independent stat.)                    '
280 '       NTTLE : total entries                                           '
290 '         NIQ : current contents (dependent stat.)                      '
300 '        DNIQ : current contents (independent stat.)                    '
310 '                                                                       '
320 '*************************************************************************
330 '
340 DEFINT I,N,Q
350 DIM MT3(4),MT4A(4,4),MT4B(4),MT5A(4),MT6A(66),MT6B(66)
360 DIM IX(4),IY(4),IZ(4),LX(4),LY(4),LZ(4),LH(4),WH(4),HT(4)
370 DIM IP(5),P1(5),P2(4,5),PU(5),PD(5),SA(4,3),JK(10,5)
380 DIM PALLET(4),NOPRE(5,66),INDEX(5,66)
390 DIM INVR(5,66),PROB(4),RANGE(20,4),NSEQ(200)
400 DIM TYPE(66),T(66),B(66),PNT(66),START(5,5)
410 DIM PX(66),PY(66),PZ(66),PO(66)
420 DIM SPP$(6,2),SPN$(6,2),MTR$(4),SIGN$(4),C(5)
430 DIM HO(3),IDX(3),MO$(3),SO$(3),STOVFL(4),QN(5)
440 DIM NIQ(4),CTM1(4),CTM2(4),DCTM(4),NMAX1(4),NMAX2(4),DNMAX(4)
450 DIM DNIQ(4),NTTLE1(4),NTTLE2(4),NGES1(4),NGES2(4),DSEQ(20)
460 '
470 '          keyboard manipulation
480 '
490 SPP$(1,1)="F+1" : SPP$(1,2)="F+10"
500 SPN$(1,1)="F-1" : SPN$(1,2)="F-10"
```

```
510 SPP$(2,1)="E+1" : SPP$(2,2)="E+10"
520 SPN$(2,1)="E-1" : SPN$(2,2)="E-10"
530 SPP$(3,1)="D+1" : SPP$(3,2)="D+10"
540 SPN$(3,1)="D-1" : SPN$(3,2)="D-10"
550 SPP$(4,1)="G+1" : SPP$(4,2)="G+10"
560 SPN$(4,1)="G-1" : SPN$(4,2)="G-10"
570 SPP$(5,1)="A+1" : SPP$(5,2)="A+5"
580 SPN$(5,1)="A-1" : SPN$(5,2)="A-5"
590 SPP$(6,1)="H+1" : SPP$(6,2)="H+10"
600 SPN$(6,1)="H-1" : SPN$(6,2)="H-10"
610 COMFIL$="COM1:9600,E,7,2,DS" : ' RS-232C communications
620 CLOSE #1
630 OPEN COMFIL$ FOR OUTPUT AS #1
640 CLS : LOCATE 10,1
650 PRINT "MANUAL OPERATION"
660 PRINT
670 PRINT "     BASE   SHLDR   ELBOW   PITCH   ROLL   TABLE"
680 PRINT "       1       2       3       4      5       6  "
690 PRINT
700 PRINT "         Q       W       E       R      T       Y  "
710 PRINT
720 PRINT "     PRESS 0--SLOW SPEED"
730 PRINT "           9--FAST SPEED"
740 PRINT : PRINT "     PRESS X TO EXIT"
750 X$=INKEY$
760 IF X$="" GOTO 750
770 IF X$="9" THEN INDEX=2
780 IF X$="0" THEN INDEX=1
790 IF X$="X" GOTO 1040
800 IF X$ <> "1" GOTO 820
810  PRINT #1,SPP$(1,INDEX) : GOTO 750
820 IF X$ <> "Q" GOTO 840
830  PRINT #1,SPN$(1,INDEX) : GOTO 750
840 IF X$ <> "2" GOTO 860
850  PRINT #1,SPP$(2,INDEX) : GOTO 750
860 IF X$ <> "W" GOTO 880
870  PRINT #1,SPN$(2,INDEX) : GOTO 750
880 IF X$ <> "3" GOTO 900
890  PRINT #1,SPP$(3,INDEX) : GOTO 750
900 IF X$ <> "E" GOTO 920
910  PRINT #1,SPN$(3,INDEX) : GOTO 750
920 IF X$ <> "4" GOTO 940
930  PRINT #1,SPP$(4,INDEX) : GOTO 750
940 IF X$ <> "R" GOTO 960
950  PRINT #1,SPN$(4,INDEX) : GOTO 750
960 IF X$ <> "5" GOTO 980
970  PRINT #1,SPP$(5,INDEX) : GOTO 750
980 IF X$ <> "T" GOTO 1000
990  PRINT #1,SPN$(5,INDEX) : GOTO 750
1000 IF X$ <> "6" GOTO 1020
```

```
1010  PRINT #1,SPP$(6,INDEX) : GOTO 750
1020 IF X$ <> "Y" GOTO 750
1030  PRINT #1,SPN$(6,INDEX) : GOTO 750
1040 CLS
1050 '
1060 '          set up parameters
1070 '
1075 NPTTN=20
1080 FOR I=1 TO NPTTN
1090 PRINT "ENTER DIST. NUMBER OF SEQUENCE ";I : INPUT DSEQ(I)
1100 NEXT I
1110 INPUT "ENTER LENGTH OF A DIST: ",QO : Q1=QO
1120 INPUT "TOTAL NUMBER OF PALLETS: ";TNP
1130 ANGL=360/TNP
1140 MTR$(1)="F" : MTR$(2)="E" : MTR$(3)="D" : MTR$(4)="A"
1150 INDEX=1 : ZO$="6" : Z1=6 : Z2=95-Z1
1160 CNYH=3.75 : STORAGEH=0! : TABLEH=7!
1170 H=10.8 : L=9 : LL=6.25
1180 PI=3.14159 : C=180!/PI : P=-90/C : R=0
1190 SF=2620*C/360 : SE=3144*C/360 : SD=SE
1200 SC=4541.3*C/360 : SA=4.1667*C
1210 FOR I=1 TO 4 : STOVFL(I)=0 : NEXT I
1220 NDIST=0 : CCT2=0 : NCLD2=0 : CPRTM=0 : OPTS2=0
1230 DATE$="1-1-1986" : TIME$="00:00:00"
1240 FOR J=1 TO 4
1250 NIQ(J)=0 : CTM2(J)=0 : NMAX2(J)=0 : NTTLE2(J)=0 : NGES2(J)=0
1260 NEXT J
1270 HZ=12.5 : AF=1.5 : NN=1 : PRCTG=0!
1280 '
1290 '                box dimensions and
1300 '                initial and extreme positions of storages
1310 '
1320 WH(1)=1.2 : LH(1)=1.2 : HT(1)=1
1330 WH(2)=2.2 : LH(2)=1.2 : HT(2)=1
1340 WH(3)=2.2 : LH(3)=2.2 : HT(3)=2
1350 WH(4)=3.2 : LH(4)=2.2 : HT(4)=1
1355 '
1360 IX(1)=9.5 : IY(1)=6 : IZ(1)=STORAGEH+1
1370 IX(2)=12.5 : IY(2)=6 : IZ(2)=STORAGEH+1
1380 IX(3)=5 : IY(3)=10 : IZ(3)=STORAGEH+2
1390 IX(4)=1.5 : IY(4)=10 : IZ(4)=STORAGEH+1
1400 LX(1)=10.7 : LY(1)=10.8 : LZ(1)=STORAGEH+4
1410 LX(2)=12.5 : LY(2)=10.8 : LZ(2)=STORAGEH+4
1420 LX(3)=7.2 : LY(3)=12.2 : LZ(3)=STORAGEH+8
1430 LX(4)=1.5 : LY(4)=12.2 : LZ(4)=STORAGEH+4
1440 FOR J=1 TO 4
1450 SA(J,1)=IX(J) : SA(J,2)=IY(J) : SA(J,3)=IZ(J)
1460 NEXT J
1470 '
1480 LNGTH=INT(QO/200) : LGTH1=QO-LNGTH*200
```

```
1490 IF LNGTH*200 <> QO THEN LNGTH=LNGTH+1
1500 FOR J=1 TO LNGTH
1510 QN(J)=200 : IF J=LNGTH AND LGTH1 > 0 THEN QN(J)=LGTH1
1520 NEXT J
1530 '
1540 '                robot movement times (in seconds)
1550 '
1560 MT1=2 : MT2=3 : MT5B=2 : MT7=3 : MT8=3
1570 MT3(1)=6 : MT3(2)=6 : MT3(3)=7 : MT3(4)=9
1580 MT4A(1,2)=6 : MT4A(1,3)=6 : MT4A(1,4)=7
1590 MT4A(2,1)=6 : MT4A(2,3)=6 : MT4A(2,4)=8
1600 MT4A(3,1)=6 : MT4A(3,2)=6 : MT4A(3,4)=5
1610 MT4A(4,1)=7 : MT4A(4,2)=8 : MT4A(4,3)=5
1620 MT4B(1)=7 : MT4B(2)=7 : MT4B(3)=8 : MT4B(4)=9
1630 MT5A(1)=2 : MT5A(2)=2 : MT5A(3)=4 : MT5A(4)=5
1640 '
1650 '                pick-up positions of various box types
1660 '
1670 X=9 : Y=0 : Z=13.55 : GOSUB 4970 : ' home position
1680 IP(1)=T1 : IP(2)=T2 : IP(3)=T3 : IP(4)=T4 : IP(5)=ANGLE
1690 X=12 : Y=-2 : Z=12 : GOSUB 4970  : ' pt. above pick-up pos.
1700 P1(1)=T1 : P1(2)=T2 : P1(3)=T3 : P1(4)=T4 : P1(5)=ANGLE
1710 X=11.5 : Y=-1.5 : Z=CNYH+1 : GOSUB 4970 : ' pos. of type 1
1720 P2(1,1)=T1: P2(1,2)=T2: P2(1,3)=T3: P2(1,4)=T4: P2(1,5)=ANGLE
1730 X=12 : Y=-1.5 : Z=CNYH+1 : GOSUB 4970 : ' pos. of type 2
1740 P2(2,1)=T1: P2(2,2)=T2: P2(2,3)=T3: P2(2,4)=T4: P2(2,5)=ANGLE
1750 X=12 : Y=-1 : Z=CNYH+2 : GOSUB 4970 : ' pos. of type 3
1760 P2(3,1)=T1: P2(3,2)=T2: P2(3,3)=T3: P2(3,4)=T4: P2(3,5)=ANGLE
1770 X=12.5 : Y=-1 : Z=CNYH+1 :GOSUB 4970 : ' pos. of type 4
1780 P2(4,1)=T1: P2(4,2)=T2: P2(4,3)=T3: P2(4,4)=T4: P2(4,5)=ANGLE
1790 FOR J=1 TO 5 : C(J)=P1(J)-IP(J) : NEXT J
1800 C(5)=ABS(C(5)) : GOSUB 5200
1810 DT=MT1 : GOSUB 7450
1820 '
1830 '                box proportions
1840 '
1850 RANGE(1,1)=0 : RANGE(1,2)=1/3 : RANGE(1,3)=1/3 : RANGE(1,4)=1/3
1860 RANGE(2,1)=0 : RANGE(2,2)=1/3 : RANGE(2,3)=2/3 : RANGE(2,4)=0
1870 RANGE(3,1)=1/3 : RANGE(3,2)=0 : RANGE(3,3)=1/3 : RANGE(3,4)=1/3
1880 RANGE(4,1)=0 : RANGE(4,2)=2/3 : RANGE(4,3)=1/3 : RANGE(4,4)=0
1890 RANGE(5,1)=1/3 : RANGE(5,2)=1/3 : RANGE(5,3)=1/3 : RANGE(5,4)=0
1900 RANGE(6,1)=0 : RANGE(6,2)=2/3 : RANGE(6,3)=0 : RANGE(6,4)=1/3
1910 RANGE(7,1)=1/3 : RANGE(7,2)=0 : RANGE(7,3)=2/3 : RANGE(7,4)=0
1920 RANGE(8,1)=1/3 : RANGE(8,2)=1/3 : RANGE(8,3)=0 :RANGE(8,4)=1/3
1930 RANGE(9,1)=0 : RANGE(9,2)=1/3 : RANGE(9,3)=0 : RANGE(9,4)=2/3
1940 RANGE(10,1)=0: RANGE(10,2)=0: RANGE(10,3)=1/3: RANGE(10,4)=2/3
1950 RANGE(11,1)=1/3: RANGE(11,2)=2/3: RANGE(11,3)=0: RANGE(11,4)=0
1960 RANGE(12,1)=0: RANGE(12,2)=0: RANGE(12,3)=0: RANGE(12,4)=1
1970 RANGE(13,1)=1/3: RANGE(13,2)=0: RANGE(13,3)=0: RANGE(13,4)=2/3
1980 RANGE(14,1)=0: RANGE(14,2)=0: RANGE(14,3)=1: RANGE(14,4)=0
```

```
1990 RANGE(15,1)=0: RANGE(15,2)=1: RANGE(15,3)=0: RANGE(15.4)=0
2000 RANGE(16,1)=0: RANGE(16,2)=0: RANGE(16,3)=2/3: RANGE(16,4)=1/3
2010 RANGE(17,1)=2/3: RANGE(17,2)=1/3: RANGE(17,3)=0: RANGE(17,4)=0
2020 RANGE(18,1)=1: RANGE(18,2)=0: RANGE(18,3)=0: RANGE(18,4)=0
2030 RANGE(19,1)=2/3: RANGE(19,2)=0: RANGE(19,3)=1/3: RANGE(20,4)=0
2040 RANGE(20,1)=2/3: RANGE(20,2)=0: RANGE(20,3)=0: RANGE(20,4)=1/3
2050 '
2060 PR=1 : TCNT=0 : F1$="PLT" : F3$=".DAT"
2070 FILEN=TNP : GOSUB 6370
2080 CLOSE #4 : OPEN "OUTPUT.DAT" FOR OUTPUT AS #4
2090 CLOSE #5 : OPEN "QUEUETS.DAT" FOR OUTPUT AS #5
2100 CLOSE #6 : OPEN "QUEUEDT.DAT" FOR OUTPUT AS #6
2110 PRINT #4,"TOTAL NUMBER OF PALLETS: ";TNP
2120 GOSUB 6500 : OLDTM2=DNEWTM : STM2=DNEWTM
2130 '
2140 '          read input data of a pallet pattern
2150 '
2160 NDIST=NDIST+1 : FILEN=DSEQ(NDIST) : GOSUB 6370
2170 CLOSE#3 : OPEN PATTERN$ FOR INPUT AS #3
2180 CURRQ=1 : Q11=QN(1) : FOR J=1 TO Q11 : NSEQ(J)=J : NEXT J
2190 GOSUB 6500 : OLDTM1=DNEWTM : STM1=DNEWTM
2200 INPUT #3,TYPE,NODE,ACT
2210 E1=0
2220 FOR J=1 TO TYPE
2230 E1=E1+RANGE(FILEN,J) : PROB(J)=E1*Q11
2240 NEXT J
2250 FOR I=1 TO NODE
2260 INPUT #3,TE,TYPE(I),PX(I),PY(I),PZ(I),PO(I),MT6A(I),MT6B(I)
2270 NEXT I
2280 FOR I=1 TO ACT : INPUT #3,T(I),B(I) : NEXT I
2290 FOR I=1 TO 5 : FOR J=1 TO 5 : START(I,J)=0 : NEXT J : NEXT I
2300 MAX=NODE : IF MAX < ACT THEN MAX=ACT
2310 '
2320 '         construct chains
2330 '
2340 FOR I=1 TO NODE : PNT(I)=0 : NEXT I
2350 FOR I=1 TO TNP+1 : FOR J=1 TO MAX
2360 NOPRE(I,J)=0 : INDEX(I,J)=0 : INVR(I,J)=0
2370 NEXT J : NEXT I
2380 B(ACT+1)=0 : T(ACT+1)=0 : J=1
2390 NOPRE(1,B(1))=NOPRE(1,B(1))+1
2400 FOR I=2 TO ACT+1
2410 NOPRE(1,B(I))=NOPRE(1,B(I))+1
2420 IF NO=T(I) GOTO 2440
2430 PNT(NO)=J : NO=T(I) : J=I
2440 NEXT I
2450 FOR I=1 TO TYPE : T(I)=0 : NEXT I
2460 FOR I=1 TO NODE
2470 IF T(TYPE(I))=0 THEN START(1,TYPE(I))=I
2480 INDEX(1,T(TYPE(I)))=I : T(TYPE(I))=I
```

```
2490 NEXT I
2500 FOR I=1 TO TYPE : T(I)=0 : NEXT I
2510 FOR I=NODE TO 1 STEP -1
2520 INVR(1,T(TYPE(I)))=I : T(TYPE(I))=I
2530 NEXT I  :  INDEX(1,0)=0
2540 FOR I=2 TO TNP+1 : FOR J=0 TO MAX
2550 NOPRE(I,J)=NOPRE(1,J) : INDEX(I,J)=INDEX(1,J)
2560 INVR(I,J)=INVR(1,J)
2570 NEXT J : NEXT I
2580 FOR I=2 TO TNP+1 : FOR J=1 TO TYPE
2590 START(I,J)=START(1,J)
2600 NEXT J : NEXT I
2610 FOR I=1 TO TNP : PALLET(I)=NODE : NEXT I
2620 '
2630 CCT1=0 : NCLD1=0 : OPTS1=0
2640 FOR J=1 TO 4
2650 NMAX1(J)=NIQ(J) : CTM1(J)=0 : NTTLE1(J)=0 : NGES1(J)=0
2660 DCTM(J)=0 : DNMAX(J)=0 : DNIQ(J)=0
2670 NEXT
2680 GOSUB 6500 : CYCTM=DNEWTM
2690 '
2700 '          pick up a box from in-feeding conveyor
2710 '
2720 GOSUB 6500 : CCT1=CCT1+DNEWTM-CYCTM
2730 CCT2=CCT2+DNEWTM-CYCTM : CYCTM=DNEWTM
2740 OPT=DNEWTM : INDXP=0
2750 GOSUB 6580 : ' generate box type
2760 FOR J=1 TO 5 : C(J)=P2(BT,J)-P1(J) : NEXT J
2770 C(5)=ABS(C(5)) : GOSUB 5200
2780 'PRINT #1,"C+09" : ' actuate the gripper
2790 GOSUB 6450
2800 FOR J=1 TO 4 : C(J)=-C(J) : NEXT J : GOSUB 5200
2810 DT=MT2 : GOSUB 7450
2820 '
2830 '          determine where to place the box
2840 '          NC=0 for pallet; NC=1 for storage area
2850 '
2860 GOSUB 4070
2870 IF NC=0 GOTO 3970
2880 IF NC=1 GOTO 2910
2890 PRINT "ERROR" : STOP
2900 '
2910 '          place box into storage area
2920 '
2930 ID$="S" : SAN=BT
2940 X=SA(BT,1) : Y=SA(BT,2) : Z=HZ : GOSUB 4970
2950 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
2960 FOR J=1 TO 5 : C(J)=PU(J)-P1(J) : NEXT J
2970 C(5)=ABS(C(5)) : GOSUB 5200
2980 X=SA(BT,1) : Y=SA(BT,2) : Z=SA(BT,3)+AF : GOSUB 4970
```

```
2990 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
3000 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
3010 C(5)=ABS(C(5)) : GOSUB 5200
3020 FOR J=1 TO 5 : T(J)=PD(J) : NEXT J
3030 TX=SA(BT,1) : TY=SA(BT,2) : TZ=SA(BT,3)
3040 GOSUB 5830 : ' straight down
3050 'PRINT #1,"CX" : ' release the gripper
3060 GOSUB 6450
3070 GOSUB 6500 : CTM1(BT)=CTM1(BT)+(DNEWTM-OLDTM1)*NIQ(BT)
3080 CTM2(BT)=CTM2(BT)+(DNEWTM-OLDTM2)*NIQ(BT)
3090 DCTM(BT)=DCTM(BT)+(DNEWTM-OLDTM1)*DNIQ(BT)
3100 DNIQ(BT)=DNIQ(BT)+1 : NIQ(BT)=NIQ(BT)+1
3110 OLDTM1=DNEWTM : OLDTM2=DNEWTM
3120 GOSUB 5990 : ' move straight up
3130 IF NMAX1(BT) < NIQ(BT) THEN NMAX1(BT)=NIQ(BT)
3140 IF DNMAX(BT) < DNIQ(BT) THEN DNMAX(BT)=DNIQ(BT)
3150 IF NMAX2(BT) < NIQ(BT) THEN NMAX2(BT)=NIQ(BT)
3160 NTTLE1(BT)=NTTLE1(BT)+1 : NTTLE2(BT)=NTTLE2(BT)+1
3170 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
3180 C(5)=ABS(C(5)) : GOSUB 5200
3190 DT=MT3(BT) : GOSUB 7450
3200 '
3210 '                  update next placement location in storage
3220 '
3230 SA(BT,1)=SA(BT,1)+WH(BT)
3240 IF SA(BT,1) <= LX(BT) GOTO 3420
3250 SA(BT,1)=IX(BT) : SA(BT,2)=SA(BT,2)+LH(BT)
3260 IF SA(BT,2) <= LY(BT) GOTO 3420
3270 SA(BT,2)=IY(BT) : SA(BT,3)=SA(BT,3)+HT(BT)
3280 IF SA(BT,3) <= LZ(BT) GOTO 3420
3290 '
3300 '                  storage overflow
3310 '                  (infinite storage capacity is assumed)
3320 '
3330 'PRINT "STORAGE OVERFLOW" : PRINT : PRINT "CLEAN THE STORAGE"
3340 'FOR J=1 TO 3 : PLAY "C+L2" : NEXT J
3350 'PRINT : PRINT "ENTER ANY KEY TO CONTINUE " : INPUT KY$ : CLS
3360 SA(BT,1)=IX(BT) : SA(BT,2)=IY(BT) : SA(BT,3)=IZ(BT)
3370 STOVFL(BT)=STOVFL(BT)+1
3380 'FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
3390 'C(5)=ABS(C(5)) : GOSUB 5200
3400 'GOTO 2700 : ' go to pick-up position
3410 '
3420 '          move one box from every storage area and
3430 '          place it onto the pallet
3440 '
3450 PR1=0
3460 ICUM=0
3470 FOR N=1 TO TYPE
3480 IF RANGE(FILEN,N)=0 THEN ICUM=ICUM+1 : GOTO 3860
```

```
3490 'IF SA(N,1)=IX(N) AND SA(N,2)=IY(N) AND SA(N,3)=IZ(N) THEN
                        ICUM=ICUM+1 : GOTO 3860
3510 IF NIQ(N)=0 THEN ICUM=ICUM+1 : GOTO 3860
3520 BT=N : GOSUB 4070
3530 IF NC=1 THEN ICUM=ICUM+1 : GOTO 3860 : ' no pallet space avail.
3540 SA(N,1)=SA(N,1)-WH(N)
3550 IF SA(N,1) >= IX(N) GOTO 3610
3560 SA(N,1)=LX(N) : SA(N,2)=SA(N,2)-LH(N)
3570 IF SA(N,2) >= IY(N) GOTO 3610
3580 SA(N,2)=LY(N) : SA(N,3)=SA(N,3)-HT(N)
3590 IF SA(N,3) >= IZ(N) GOTO 3610
3600 SA(N,1)=IX(N) : SA(N,2)=IY(N) : SA(N,3)=IZ(N)
3610 X=SA(N,1) : Y=SA(N,2) : Z=HZ : GOSUB 4970
3620 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
3630 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
3640 C(5)=ABS(C(5)) : GOSUB 5200
3650 X=SA(N,1) : Y=SA(N,2) : Z=SA(N,3)+AF : GOSUB 4970
3660 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
3670 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
3680 C(5)=ABS(C(5)) : GOSUB 5200
3690 FOR J=1 TO 5 : T(J)=PU(J) : NEXT J
3700 TX=SA(N,1) : TY=SA(N,2) : TZ=SA(N,3) : GOSUB 5830
3710 'PRINT #1,"C+9"
3720 GOSUB 6450 : GOSUB 5990
3730 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
3740 C(5)=ABS(C(5)) : GOSUB 5200
3750 IF ID$="S" THEN DT=MT4A(SAN,BT) ELSE DT=MT4B(BT)
3760 GOSUB 7450
3770 GOSUB 6500
3780 CTM1(BT)=CTM1(BT)+(DNEWTM-OLDTM1)*NIQ(BT)
3790 CTM2(BT)=CTM2(BT)+(DNEWTM-OLDTM2)*NIQ(BT)
3800 DCTM(BT)=DCTM(BT)+(DNEWTM-OLDTM1)*DNIQ(BT)
3810 NIQ(BT)=NIQ(BT)-1 : DNIQ(BT)=DNIQ(BT)-1
3820 IF DNIQ(BT) < 0 THEN DNIQ(BT)=0
3830 OLDTM1=DNEWTM : OLDTM2=DNEWTM : INDXP=0
3840 IDX$="SA" :SAN=BT
3850 GOSUB 4420 : ' place box onto pallet
3860 NEXT N
3870 IF ICUM <> TYPE AND PALLET(PR)/NODE <= PRCTG THEN
                    PR1=1 : GOTO 3460
3890 GOSUB 6500 : OPT=DNEWTM
3900 FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
3910 C(5)=ABS(C(5)) : GOSUB 5200
3920 IF ID$="S" THEN DT=MT5A(SAN) ELSE DT=MT5B
3930 GOSUB 7450
3940 GOSUB 6500 : OPTS1=OPTS1+DNEWTM-OPT : OPTS2=OPTS2+DNEWTM-OPT
3950 GOTO 2700 : ' go to pick-up position
3960 '
3970 '         the pallet
3980 '
```

```
3990 INDXP=1 : IDX$="P1"
4000 FOR J=1 TO 5 : PD(J)=P1(J) : NEXT J
4010 GOSUB 4420 : ' place box onto pallet
4020 GOSUB 6500
4030 OPTS1=OPTS1+DNEWTM-OPT : OPTS2=OPTS2+DNEWTM-OPT : INDXP=0
4040 GOTO 3420 : ' move one box from every storage
4050 END
4060 '
4070 '        search chain to determine box's placement location
4080 '
4090 T(1)=PR : TE=PR : NC=0
4100 IF TNP=1 GOTO 4140
4110 FOR I=2 TO TNP
4120 TE=TE+1 : IF TE > TNP THEN TE=1
4130 T(I)=TE : NEXT I
4140 FOR K=1 TO TNP : TE=T(K) : SEQ=K
4150 SRCH=START(TE,BT)
4160 FOR I=1 TO NODE
4170 IF START(TE,BT)=0 GOTO 4230
4180 IF NOPRE(TE,SRCH)=0 THEN GOTO 4260
4190 SRCH=INDEX(TE,SRCH)
4200 IF SRCH=0 GOTO 4230
4210 NEXT I
4220 IF PR1=1 GOTO 4240
4230 NEXT K
4240 NC=1 : RETURN : ' no pallet space available
4250 '
4260 '                update the chain
4270 '
4280 IF START(TE,BT)=SRCH THEN
                    START(TE,BT)=INDEX(TE,SRCH) : GOTO 4320
4300 INDEX(TE,INVR(TE,SRCH))=INDEX(TE,SRCH)
4310 INVR(TE,INDEX(TE,SRCH))=INVR(TE,SRCH)
4320 IF PNT(SRCH)=0 GOTO 4400
4330 NXT=SRCH : FOR I=1 TO NODE : NXT=NXT+1
4340 IF NXT=NODE+1 THEN NO=ACT : GOTO 4370
4350 IF PNT(NXT) <> 0 THEN NO=PNT(NXT)-1 : GOTO 4370
4360 NEXT I : PRINT "ERROR ON PNT" : STOP
4370 FOR K=PNT(SRCH) TO NO
4380 NOPRE(TE,B(K))=NOPRE(TE,B(K))-1
4390 NEXT K
4400 PLTN=TE : NC=0 : RETURN
4410 '
4420 '        place boxes onto pallet
4430 '
4440 ID$="P"
4450 IF ANGL*(SEQ-1) <= 180 GOTO 4470
4460 SPN$(1,1)="-" : RTH=(360/ANGL-SEQ+1)*2620/TNP : GOTO 4480
4470 SPN$(1,1)="+" : RTH=2620/TNP*(SEQ-1)
4480 GOSUB 6070 : ' rotate table for desired pallet
```

```
4490 X=PX(SRCH) : Y=PY(SRCH) : Z=HZ : GOSUB 4970
4500 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
4510 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
4520 C(5)=ABS(C(5)) : GOSUB 5200
4530 IF PO(SRCH) <> 1 GOTO 4550
4540 SPN$(2,2)="+" : GOSUB 6220
4550 X=PX(SRCH) : Y=PY(SRCH) : Z=PZ(SRCH)+TABLEH+AF : GOSUB 4970
4560 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
4570 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
4580 C(5)=ABS(C(5)) : GOSUB 5200
4590 FOR J=1 TO 5 : T(J)=PD(J) : NEXT J
4600 TX=PX(SRCH) : TY=PY(SRCH) : TZ=PZ(SRCH)+TABLEH : GOSUB 5830
4610 'PRINT #1,"CX"
4620 GOSUB 6450 : GOSUB 5990
4630 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
4640 C(5)=ABS(C(5)) : GOSUB 5200
4650 IF IDX$="SA" THEN DT=MT6B(SRCH) ELSE DT=MT6A(SRCH)
4660 GOSUB 7450
4670 IF SPN$(1,1)="+" THEN SPN$(1,1)="-" ELSE SPN$(1,1)="+"
4680 GOSUB 6070 : ' rotate table back to original pallet
4690 PALLET(PLTN)=PALLET(PLTN)-1 : NCLD1=NCLD1+1 : NCLD2=NCLD2+1
4700 IF PO(SRCH) <> 1 GOTO 4720
4710 SPN$(2,2)="-" : GOSUB 6220
4720 IF PALLET(PLTN) <> 0 THEN RETURN
4730 '
4740 '                pallet is full
4750 '
4760 FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
4770 C(5)=ABS(C(5)) : GOSUB 5200
4780 DT=MT5B : GOSUB 7450
4790 IF PR=TNP THEN PR=1 ELSE PR=PR+1
4800 NO=TNP+1
4810 FOR J=0 TO MAX
4820 NOPRE(PLTN,J)=NOPRE(NO,J) : INDEX(PLTN,J)=INDEX(NO,J)
4830 INVR(PLTN,J)=INVR(NO,J)
4840 NEXT J
4850 FOR J=1 TO TYPE : START(PLTN,J)=START(NO,J) : NEXT J
4860 PALLET(PLTN)=NODE
4870 RTH=655
4880 SPN$(1,1)="+" : GOSUB 6070 : ' rotate turntable
4890 'PRINT "REMOVE THE PALLET AND INSERT A NEW PALLET" : PRINT
4900 'PRINT "ENTER ANY KEY WHEN READY" : INPUT SPP$(1,1) : CLS
4910 IF INDXP=0 GOTO 2700
4920 GOSUB 6500
4930 OPTS1=OPTS1+DNEWTM-OPT : OPTS2=OPTS2+DNEWTM-OPT : INDXP=0
4940 GOTO 2700
4950 RETURN
4960 '
4970 '        coordinate transformation
4980 '
```

```
4990 RR=SQR(X*X+Y*Y)
5000 IF X = 0 THEN T1=SGN(Y)*PI/2
5010 IF X > 0 THEN T1=ATN(Y/X)
5020 IF X < 0 AND Y > 0 THEN T1=PI-ATN(Y/ABS(X))
5030 IF X < 0 AND Y < 0 THEN T1=-(PI-ATN(Y/X))
5040 ANGLE=ABS(T1*C) : IF X=0 THEN T4=0 ELSE T4=ATN(ABS(Y/X))
5050 RO=RR-LL*COS(P)
5060 ZO=Z-LL*SIN(P)-H
5070 IF RO=0 THEN G=SGN(ZO)*PI/2! ELSE G=ATN(ZO/RO)
5080 A=RO*RO+ZO*ZO
5090 A=4*L*L/A-1
5100 A=ABS(A)
5110 A=ATN(SQR(A))
5120 T2=A+G
5130 T3=G-A
5140 T1=INT(T1*SF)
5150 T2=INT(T2*SE)
5160 T3=INT(T3*SD)
5170 T4=INT(T4*SA)
5180 RETURN
5190 '
5200 '         simultaneous movements of joints
5210 '
5220 RETURN : ' this subroutine is not carried out in simulation
5230 SIGN$(1)="+" : SIGN$(2)="+" : SIGN$(3)="+" : SIGN$(4)="+"
5240 IF C(1) < 0 THEN SIGN$(1)="-"
5250 IF C(2) > 0 THEN SIGN$(2)="-"
5260 IF C(3) < 0 THEN SIGN$(3)="-"
5270 IF C(5) <= 90 AND C(1) < 0 THEN SIGN$(4)="-"
5280 IF C(5) <= 90 AND C(1) > 0 THEN SIGN$(4)="+"
5290 IF C(5) > 90 AND C(1) < 0 THEN SIGN$(4)="+"
5300 IF C(5) > 90 AND C(1) > 0 THEN SIGN$(4)="-"
5310 '
5320 HO(1)=ABS(C(1)) : HO(2)=ABS(C(2)) : HO(3)=ABS(C(3))
5330 FOR J1=1 TO 2 : L1=J1 : JJ=J1+1 : FOR J2=JJ TO 3
5340 IF HO(L1) < HO(J2) THEN L1=J2
5350 NEXT J2
5360 TE=HO(J1) : HO(J1)=HO(L1) : HO(L1)=TE
5370 TE=IDX(J1) : IDX(J1)=IDX(L1) : IDX(L1)=TE
5380 NEXT J1
5390 FOR I=1 TO 3
5400 MO$(I)=MTR$(IDX(I)) : SO$(I)=SIGN$(IDX(I))
5410 NEXT I
5420 '
5430 IF HO(1)=0 GOTO 5680
5440 E1=0 : COUNT=0 : RATE1=HO(2)/HO(1) : RATE2=HO(3)/HO(1)
5450 D1=0 : HS=HO(1)
5460 PRINT #1,MO$(1);"?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5470 IF CD > Z2 GOTO 5460
5480 IF HS > Z1 GOTO 5510
```

```
5490 TN$=CHR$(HS+48) : PRINT #1,MO$(1);SO$(1);TN$
5500 COUNT=COUNT+HS : GOTO 5530
5510 PRINT #1,MO$(1);SO$(1);ZO$
5520 COUNT=COUNT+Z1
5530 HS=HS-Z1
5540 IF HO(2)=0 GOTO 5640
5550 E2=INT(COUNT*RATE1) : TE=E2-E1
5560 IF TE=0 GOTO 5590
5570 E1=E2 : TN$=CHR$(TE+48)
5580 PRINT #1,MO$(2);SO$(2);TN$
5590 IF HO(3)=0 GOTO 5640
5600 D2=INT(COUNT*RATE2) : TE=D2-D1
5610 IF TE=0 GOTO 5640
5620 D1=D2 : TN$=CHR$(TE+48)
5630 PRINT #1,MO$(3);SO$(3);TN$
5640 IF HS > 0 GOTO 5460
5650 '
5660 '                  roll the hand
5670 '
5680 HS=ABS(C(4)) : IF HS=0 GOTO 5760
5690 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5700 IF CD > Z2 GOTO 5690
5710 IF HS > Z1 GOTO 5730
5720 TN$=CHR$(HS+48) : PRINT #1,MTR$(4);SIGN$(4);TN$ : GOTO 5760
5730 PRINT #1,MTR$(4);SIGN$(4);ZO$
5740 HS=HS-Z1
5750 IF HS > 0 GOTO 5690
5760 CD=0
5770 FOR I=1 TO 4 : PRINT #1,MTR$(I);"?"
5780 TN$=INPUT$(1,#1) : CD=CD+ASC(TN$)-32
5790 NEXT I
5800 IF CD <> 0 GOTO 5760
5810 RETURN
5820 '
5830 '         move the arm straight down
5840 '
5850 JM=AF/NN : SZ=TZ+AF
5860 FOR J=1 TO NN
5870 X=TX : Y=TY : Z=SZ-JM*J : GOSUB 4970
5880 TYPE(1)=T1 : TYPE(2)=T2 : TYPE(3)=T3
5890 TYPE(4)=T4 : TYPE(5)=ANGLE
5900 FOR JP=1 TO 5 : JK(J,JP)=TYPE(JP)-T(JP) : NEXT JP
5910 JK(J,5)=ABS(JK(J,5))
5920 FOR JP=1 TO 5 : T(JP)=TYPE(JP) : NEXT JP
5930 NEXT J
5940 FOR J=1 TO NN
5950 FOR JP=1 TO 5 : C(JP)=JK(J,JP) : NEXT JP : GOSUB 5200
5960 NEXT J
5970 RETURN
5980 '
```

```
5990 '          move the arm straight up
6000 '
6010 FOR J=NN TO 1 STEP -1
6020 FOR JP=1 TO 4 : C(JP)=-JK(J,JP) : NEXT JP
6030 C(5)=ABS(JK(J,5)) : GOSUB 5200
6040 NEXT J
6050 RETURN
6060 '
6070 '          rotate the turntable
6080 '
6090 DT=MT7 : GOSUB 7450
6100 RETURN : 'this subroutine is not carried out in simulation
6110 HS=RTH
6120 PRINT #1,"H?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6130 IF CD > 85 GOTO 6120
6140 IF HS >= 10 GOTO 6160
6150 TN$=CHR$(HS+48) : PRINT #1,"H";SPN$(1,1);TN$ : GOTO 6180
6160 PRINT #1,"H";SPN$(1,1);"10" : HS=HS-10
6170 IF HS > 0 GOTO 6120
6180 PRINT #1,"H?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6190 IF CD <> 0 GOTO 6180
6200 RETURN
6210 '
6220 '          rotate the hand to change box's orientation
6230 '
6240 DT=MT8 : GOSUB 7450
6250 RETURN : ' this subroutine is not carried out in simulation
6260 HS=375
6270 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6280 IF CD > 85 GOTO 6270
6290 IF HS >= 6 GOTO 6310
6300 TN$=CHR$(HS+48) : PRINT #1,"A";SPN$(2,2);TN$ : GOTO 6330
6310 PRINT #1,"A";SPN$(2,2);"6" : HS=HS-6
6320 IF HS > 0 GOTO 6270
6330 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6340 IF CD <> 0 GOTO 6330
6350 RETURN
6360 '
6370 '          convert number to characters for input file name
6380 '
6390 IF FILEN < 10 THEN F2$=CHR$(FILEN+48) : GOTO 6420
6400 E1=INT(FILEN/10) : R10$=CHR$(E1+48)
6410 E2=FILEN-E1*10 : R01$=CHR$(E2+48) : F2$=R10$+R01$
6420 PATTERN$=F1$+F2$+F3$
6430 RETURN
6440 '
6450 '          delay for picking up and placing a box
6460 '
6470 FOR J=1 TO 500 : E1=SQR(2) : NEXT J
6480 RETURN
```

```
6490 '
6500 '          obtain current time
6510 '
6520 SNAP1$=DATE$ : SNAP2$=TIME$
6530 DNEWTM=VAL(MID$(SNAP1$,4,2))*864001+VAL(LEFT$(SNAP2$,2))*3600!
6540 DNEWTM=DNEWTM+VAL(MID$(SNAP2$,4,2))*60!
6550 DNEWTM=DNEWTM+VAL(RIGHT$(SNAP2$,2))-86400!
6560 RETURN
6570 '
6580 '          generate box type (one box at a time)
6590 '
6600 E1=INT(RND*Q11)+1
6610 FOR J=1 TO TYPE
6620 IF NSEQ(E1)<=PROB(J) THEN BT=J : GOTO 6650
6630 NEXT J
6640 PRINT "ERROR ON BT"
6650 NGES1(BT)=NGES1(BT)+1 : NGES2(BT)=NGES2(BT)+1
6660 'FOR J=1 TO BT : PLAY "G+L6" : NEXT J
6670 'CLS : E9=350 : E2=30
6680 'FOR J=1 TO BT : CIRCLE (E9,E2),20,BT : PAINT (E9,E2),BT
6690 'E2=E2+35 : NEXT J
6700 NSEQ(E1)=NSEQ(Q11) : Q11=Q11-1
6710 IF Q11 <> 0 THEN GOTO 6780
6720 CURRQ=CURRQ+1 : IF CURRQ > LNGTH GOTO 6780
6730 Q11=QN(CURRQ) : FOR J=1 TO Q11 : NSEQ(J)=J : NEXT J
6740 E1=0
6750 FOR J=1 TO TYPE
6760 E1=E1+RANGE(FILEN,J) : PROB(J)=E1*Q11
6770 NEXT J
6780 Q1=Q1-1
6790 IF Q1 MOD 5 <> 0 OR Q1 < 5 GOTO 6860
6800 PRINT#5,"QUEUE AT ";QO-Q1;": ";NIQ(1);NIQ(2);NIQ(3);NIQ(4)
6810 PRINT#6,"QUEUE AT ";QO-Q1;": ";DNIQ(1);DNIQ(2);DNIQ(3);DNIQ(4)
6820 '
6830 '          end of a dist. run if Q1=0.  deliver statistics and
6840 '          read data of a pallet pattern for the next dist.
6850 '
6860 IF Q1 <> 0 THEN RETURN
6870 Q1=QO : GOSUB 6500 : PRTTM=DNEWTM
6880 FOR J=1 TO 4 : CTM1(J)=CTM1(J)+(DNEWTM-OLDTM1)*NIQ(J)
6890 DCTM(J)=DCTM(J)+(DNEWTM-OLDTM1)*DNIQ(J) : NEXT J
6900 '
6910 '          deliver statistics of a distribution run
6920 '
6930 PRINT#4,"***STATISTICS OF DISTRIBUTION: ";FILEN
6940 PRINT#5,"***PALLET PATTERN: ";FILEN
6950 PRINT#6,">>>PALLET PATTERN: ";FILEN
6960 PRINT#4,"TOTAL NUMBER OF CARTONS LOADED TO THE PALLET: ";NCLD1
6970 PRINT#4,"TOTAL SIMULATION TIME (DIST): ";DNEWTM-STM1
6980 PRINT#4,"AVERAGE CYCLE TIME (PICK-UP): ";CCT1/QO
```

```
6990 PRINT#4,"OPERATION TIME IN STORAGE AREA: ";DNEWTM-STM1-OPTS1
7000 FOR J=1 TO 4
7010 PRINT#4,"CUMULATIVE TIME AREA: ";J,CTM1(J),DCTM(J)
7020 AVE1=CTM1(J)/(DNEWTM-STM1) : AVE2=DCTM(J)/(DNEWTM-STM1)
7030 PRINT#4,"AVERAGE CONTENTS: ";J,AVE1;AVE2
7040 IF NTTLE1(J)=0 THEN
        PRINT#4,"AVERAGE WAITING TIME: ";J,0,0 : GOTO 7080
7060 AVE1=CTM1(J)/NTTLE1(J) : AVE2=DCTM(J)/NTTLE1(J)
7070 PRINT#4,"AVERAGE WAITING TIME: ";J,AVE1;AVE2
7080 PRINT#4,"TOTAL CARTONS ENTERED: ";J,NGES1(J)
7090 PRINT#4,"MAXIMUM CONTENTS: ";J,NMAX1(J),DNMAX(J)
7100 PRINT#4,"TOTAL ENTRIES: ";J,NTTLE1(J)
7110 PRINT#4,"ZERO ENTRIES: ";J,NGES1(J)-NTTLE1(J)
7120 PRINT#4,"CURRENT CONTENTS: ";J,NIQ(J),DNIQ(J)
7130 PRINT#4,"PROPORTIONS: ";J, RANGE(FILEN,J)
7140 NEXT J
7150 GOSUB 6500
7160 IF NDIST<>NPTTN THEN CPRTM=CPRTM+(DNEWTM-PRTTM) : GOTO 2160
7170 FOR J=1 TO 4
7180 CTM2(J)=CTM2(J)+(PRTTM-OLDTM2)*NIQ(J)-CPRTM
7190 NEXT J
7200 '
7210 '        deliver statistics of total simulation
7220 '
7230 PRINT#4,">>>STATISTICS OF TOTAL SIMULATION"
7240 PRINT#4,"TOTAL NUMBER OF CARTONS LOADED TO THE PALLET: ";NCLD2
7250 PRINT#4,"TOTAL SIMULATION TIME (ENTIRE): ";PRTTM-STM2
7260 PRINT#4,"AVERAGE CYCLE TIME (PICK-UP): ";CCT2/(QO*NPTTN)
7270 PRINT#4,"OPERATION TIME IN STORAGE AREA: ";PRTTM-STM2-OPTS2
7280 FOR J=1 TO 4
7290 PRINT#4,"CUMULATIVE TIME AREA: ";J,CTM2(J)
7300 PRINT#4,"AVERAGE CONTENTS: ";J,CTM2(J)/(PRTTM-STM2)
7310 IF NTTLE2(J)=0 THEN
        PRINT#4,"AVERAGE WAITING TIME: ";J,0,0 : GOTO 7340
7330 PRINT#4,"AVERAGE WAITING TIME: ";J,CTM2(J)/NTTLE2(J)
7340 PRINT#4,"TOTAL CARTONS ENTERED: ";J,NGES2(J)
7350 PRINT#4,"MAXIMUM CONTENTS: ";J,NMAX2(J)
7360 PRINT#4,"TOTAL ENTRIES: ";J,NTTLE2(J)
7370 PRINT#4,"ZERO ENTRIES: ";J,NGES2(J)-NTTLE2(J)
7380 PRINT#4,"NUMBER OF OVERFLOWS: ";J,STOVFL(J)
7390 PRINT#4,"CURRENT CONTENTS: ";J,NIQ(J)
7400 NEXT J
7410 CLS : PRINT "READ OUTPUT.DAT"
7420 PRINT "READ QUEUETS.DAT" : PRINT "READ QUEUEDT.DAT"
7430 END : RETURN
7440 '
7450 '        delay for robot movement time
7460 '
7470 GOSUB 6500 : DT=DNEWTM+DT
7480 GOSUB 6500 : TE=DNEWTM
```

```
7490 IF TE < DT GOTO 7480
7500 RETURN
```

XIII. APPENDIX E.

SIMULATION PROGRAM LISTING

(UNKNOWN BOX SIZE DISTRIBUTIONS)

This program is employed to simulate the robotic palletizing operations with alternate values of look-ahead factors. The length and box proportions of a distribution run cannot be determined before palletizing starts. Whenever a pallet is full, the dynamic selection procedure is executed to determine the best "match" pallet pattern according to the boxes on the in-feeding conveyor.

Definition of program variables:

FCT	= the look-ahead factor
NBOX	= cumulative number of boxes placed of a distribution run
TVOL	= cumulative box volumes in the look-ahead queue
TBOX	= cumulative number of boxes of all sizes in the look-ahead queue
NTYPE(i,j)	= number of boxes of size j in pallet pattern #i
QTYPE(j)	= cumulative number of boxes of size j in the look-ahead queue
BRATIO(i)	= cumulative deviations of box ratio for pallet pattern #i
CHECK(i)	= group number of pallet pattern #i
NBSEQ(i)	= sequence of box size numbers generated by the random number generator

The remaining variables have been defined in Appendixes C and D.

```
10 '*************************************************************************
20 '                                                                        '
30 '     Purpose - Robotic palletizing simulation program                   '
40 '               for determining look-ahead factor                        '
50 '               with unknown distributions                               '
60 '               (dynamic selection procedure for a best                  '
70 '               matched pallet pattern)                                  '
80 '                                                                        '
90 '     Parameters - see the definition in the previous                     '
100 '                  program                                               '
110 '                                                                        '
120 '*************************************************************************
130 '
140 DEFINT I,N,Q
150 DIM QTYPE(20),NBSEQ(2,200),NTYPE(20,4)
160 DIM MT3(4),MT4A(4,4),MT4B(4),MT5A(4),MT6A(66),MT6B(66)
170 DIM BV(4),CHECK(20),BRATIO(20),NSEQ(200),RANGE(20,4)
180 DIM IX(4),IY(4),IZ(4),LX(4),LY(4),LZ(4),LH(4),WH(4),HT(4)
190 DIM IP(5),P1(5),P2(4,5),PU(5),PD(5),SA(4,3),JK(10,5)
200 DIM PALLET(4),NOPRE(5,66),INDEX(5,66),INVR(5,66),PROB(4)
210 DIM TYPE(66),T(66),B(66),PNT(66),START(5,5),STOVFL(4)
220 DIM PX(66),PY(66),PZ(66),PO(66)
230 DIM SPP$(6,2),SPN$(6,2),MTR$(4),SIGN$(4)
240 DIM C(5),HO(3),IDX(3),MO$(3),SO$(3)
250 DIM NIQ(4),CTM1(4),CTM2(4),DCTM(4),NMAX1(4),NMAX2(4),DNMAX(4)
260 DIM DNIQ(4),NTTLE1(4),NTTLE2(4),NGES1(4),NGES2(4),DSEQ(20)
270 '
280 '         keyboard manipulation
290 '
300 SPP$(1,1)="F+1" : SPP$(1,2)="F+10"
310 SPN$(1,1)="F-1" : SPN$(1,2)="F-10"
320 SPP$(2,1)="E+1" : SPP$(2,2)="E+10"
330 SPN$(2,1)="E-1" : SPN$(2,2)="E-10"
340 SPP$(3,1)="D+1" : SPP$(3,2)="D+10"
350 SPN$(3,1)="D-1" : SPN$(3,2)="D-10"
360 SPP$(4,1)="G+1" : SPP$(4,2)="G+10"
370 SPN$(4,1)="G-1" : SPN$(4,2)="G-10"
380 SPP$(5,1)="A+1" : SPP$(5,2)="A+5"
390 SPN$(5,1)="A-1" : SPN$(5,2)="A-5"
400 SPP$(6,1)="H+1" : SPP$(6,2)="H+10"
410 SPN$(6,1)="H-1" : SPN$(6,2)="H-10"
420 COMFIL$="COM1:9600,E,7,2,DS"
430 CLOSE #1 : OPEN COMFIL$ FOR OUTPUT AS #1
440 CLS : LOCATE 10,1
450 PRINT "MANUAL OPERATION"
460 PRINT
470 PRINT "     BASE    SHLDR    ELBOW    PITCH    ROLL    TABLE"
480 PRINT "       1        2        3        4       5        6  "
490 PRINT
500 PRINT "        Q        W        E        R       T        Y  "
```

```
510 PRINT
520 PRINT "      PRESS 0--SLOW SPEED"
530 PRINT "            9--FAST SPEED"
540 PRINT : PRINT "     PRESS X TO EXIT"
550 X$=INKEY$
560 IF X$="" GOTO 550
570 IF X$="9" THEN INDEX=2
580 IF X$="0" THEN INDEX=1
590 IF X$="X" GOTO 870
600 IF X$ <> "1" GOTO 620
610  PRINT #1,SPP$(1,INDEX) : GOTO 550
620 IF X$ <> "Q" GOTO 640
630  PRINT #1,SPN$(1,INDEX) : GOTO 550
640 IF X$ <> "2" GOTO 660
650  PRINT #1,SPP$(2,INDEX) : GOTO 550
660 IF X$ <> "W" GOTO 680
670  PRINT #1,SPN$(2,INDEX) : GOTO 550
680 IF X$ <> "3" GOTO 700
690  PRINT #1,SPP$(3,INDEX) : GOTO 550
700 IF X$ <> "E" GOTO 720
710  PRINT #1,SPN$(3,INDEX) : GOTO 550
720 IF X$ <> "4" GOTO 740
730  PRINT #1,SPP$(4,INDEX) : GOTO 550
740 IF X$ <> "R" GOTO 760
750  PRINT #1,SPN$(4,INDEX) : GOTO 550
760 IF X$ <> "5" GOTO 780
770  PRINT #1,SPP$(5,INDEX) : GOTO 550
780 IF X$ <> "T" GOTO 800
790  PRINT #1,SPN$(5,INDEX) : GOTO 550
800 IF X$ <> "6" GOTO 820
810  PRINT #1,SPP$(6,INDEX) : GOTO 550
820 IF X$ <> "Y" GOTO 550
830  PRINT #1,SPN$(6,INDEX) : GOTO 550
840 '
850 '          set up parameters
860 '
870 CLS : NPTTN=20
880 FOR I=1 TO NPTTN
890 PRINT "ENTER DIST. NUMBER OF SEQUENCE ";I : INPUT DSEQ(I)
900 NEXT I
910 INPUT "ENTER LOOK-AHEAD FACTOR ( >= 1): ",FCT
920 CLOSE #4 : OPEN "OUTPUT.DAT" FOR OUTPUT AS #4
930 CPP=0 : CNYH=3.75 : STORAGEH=0! : TABLEH=7!
940 TNP=1 : PRCTG=0!
950 MTR$(1)="F" : MTR$(2)="E" : MTR$(3)="D" : MTR$(4)="A"
960 INDEX=1 : Z0$="6" : Z1=6 : Z2=95-Z1
970 H=10.8 : L=9 : LL=6.25
980 PI=3.14159 : C=180!/PI : P=-90/C : R=0
990 SF=2620*C/360 : SE=3144*C/360 : SD=SE
1000 SC=4541.3*C/360 : SA=4.1667*C
```

```
1010 FOR I=1 TO 4 : STOVFL(I)=0 : NEXT I
1020 NDIST=1 : CCT2=0 : NCLD2=0 : CPRTM=0 : OPTS2=0
1030 DATE$="1-1-1986" : TIME$="00:00:00"
1040 FOR J=1 TO 4
1050 NIQ(J)=0 : CTM2(J)=0 : NMAX2(J)=0 : NTTLE2(J)=0 : NGES2(J)=0
1060 NEXT J
1070 HZ=12.5 : AF=1.5 : NN=1 : Q0=200 : Q1=Q0 : TYPE=4
1080 '
1090 '                    robot movement times
1100 '
1110 MT1=2 : MT2=3 : MT5B=2 : MT7=3 : MT8=3
1120 MT3(1)=6 : MT3(2)=6 : MT3(3)=7 : MT3(4)=9
1130 MT4A(1,2)=6 : MT4A(1,3)=6 : MT4A(1,4)=7
1140 MT4A(2,1)=6 : MT4A(2,3)=6 : MT4A(2,4)=8
1150 MT4A(3,1)=6 : MT4A(3,2)=6 : MT4A(3,4)=5
1160 MT4A(4,1)=7 : MT4A(4,2)=8 : MT4A(4,3)=5
1170 MT4B(1)=7 : MT4B(2)=7 : MT4B(3)=8 : MT4B(4)=9
1180 MT5A(1)=2 : MT5A(2)=2 : MT5A(3)=4 : MT5A(4)=5
1190 '
1200 '                    box dimensions
1210 '
1220 WH(1)=1.2 : LH(1)=1.2 : HT(1)=1
1230 WH(2)=2.2 : LH(2)=1.2 : HT(2)=1
1240 WH(3)=2.2 : LH(3)=2.2 : HT(3)=2
1250 WH(4)=3.2 : LH(4)=2.2 : HT(4)=1
1260 '
1270 '                  initial & extreme pos. of storage areas
1280 '
1290 IX(1)=9.5 : IY(1)=6 : IZ(1)=STORAGEH+1
1300 IX(2)=12.5 : IY(2)=6 : IZ(2)=STORAGEH+1
1310 IX(3)=5 : IY(3)=10 : IZ(3)=STORAGEH+2
1320 IX(4)=1.5 : IY(4)=10 : IZ(4)=STORAGEH+1
1330 LX(1)=10.7 : LY(1)=10.8 : LZ(1)=STORAGEH+4
1340 LX(2)=12.5 : LY(2)=10.8 : LZ(2)=STORAGEH+4
1350 LX(3)=7.2 : LY(3)=12.2 : LZ(3)=STORAGEH+8
1360 LX(4)=1.5 : LY(4)=12.2 : LZ(4)=STORAGEH+4
1370 '
1380 '                  pick-up positions
1390 '
1400 X=9 : Y=0 : Z=13.55 : GOSUB 4890 : ' home position
1410 IP(1)=T1 : IP(2)=T2 : IP(3)=T3 : IP(4)=T4 : IP(5)=ANGLE
1420 X=12 : Y=-2 : Z=12 : GOSUB 4890 : ' pt. above pick-up pos.
1430 P1(1)=T1 : P1(2)=T2 : P1(3)=T3 : P1(4)=T4 : P1(5)=ANGLE
1440 X=11.5 : Y=-1.5 : Z=CNYH+1 : GOSUB 4890 : ' pos. of type 1
1450 P2(1,1)=T1 : P2(1,2)=T2 : P2(1,3)=T3
1460 P2(1,4)=T4 : P2(1,5)=ANGLE
1470 X=12 : Y=-1.5 : Z=CNYH+1 : GOSUB 4890 : ' pos. of type 2
1480 P2(2,1)=T1 : P2(2,2)=T2 : P2(2,3)=T3
1490 P2(2,4)=T4 : P2(2,5)=ANGLE
1500 X=12 : Y=-1 : Z=CNYH+2 : GOSUB 4890 : ' pos. of type 3
```

```
1510 P2(3,1)=T1 : P2(3,2)=T2 : P2(3,3)=T3
1520 P2(3,4)=T4 : P2(3,5)=ANGLE
1530 X=12.5 : Y=-1 : Z=CNYH+1 :GOSUB 4890 : ' pos. of type 4
1540 P2(4,1)=T1 : P2(4,2)=T2 : P2(4,3)=T3
1550 P2(4,4)=T4 : P2(4,5)=ANGLE
1560 FOR J=1 TO 4
1570 SA(J,1)=IX(J) : SA(J,2)=IY(J) : SA(J,3)=IZ(J)
1580 NEXT J
1590 FOR J=1 TO 5 : C(J)=P1(J)-IP(J) : NEXT J : C(5)=ABS(C(5))
1600 GOSUB 5120 : DT=MT1 : GOSUB 7270
1610 '
1620 '                box proportions
1630 '
1640 RANGE(1,1)=0 : RANGE(1,2)=1/3 : RANGE(1,3)=1/3 : RANGE(1,4)=1/3
1650 RANGE(2,1)=0 : RANGE(2,2)=1/3 : RANGE(2,3)=2/3 : RANGE(2,4)=0
1660 RANGE(3,1)=1/3 : RANGE(3,2)=0 : RANGE(3,3)=1/3 : RANGE(3,4)=1/3
1670 RANGE(4,1)=0 : RANGE(4,2)=2/3 : RANGE(4,3)=1/3 : RANGE(4,4)=0
1680 RANGE(5,1)=1/3 : RANGE(5,2)=1/3 : RANGE(5,3)=1/3 : RANGE(5,4)=0
1690 RANGE(6,1)=0 : RANGE(6,2)=2/3 : RANGE(6,3)=0 : RANGE(6,4)=1/3
1700 RANGE(7,1)=1/3 : RANGE(7,2)=0 : RANGE(7,3)=2/3 : RANGE(7,4)=0
1710 RANGE(8,1)=1/3 : RANGE(8,2)=1/3 : RANGE(8,3)=0 :RANGE(8,4)=1/3
1720 RANGE(9,1)=0 : RANGE(9,2)=1/3 : RANGE(9,3)=0 : RANGE(9,4)=2/3
1730 RANGE(10,1)=0: RANGE(10,2)=0: RANGE(10,3)=1/3: RANGE(10,4)=2/3
1740 RANGE(11,1)=1/3: RANGE(11,2)=2/3: RANGE(11,3)=0: RANGE(11,4)=0
1750 RANGE(12,1)=0: RANGE(12,2)=0: RANGE(12,3)=0: RANGE(12,4)=1
1760 RANGE(13,1)=1/3: RANGE(13,2)=0: RANGE(13,3)=0: RANGE(13,4)=2/3
1770 RANGE(14,1)=0: RANGE(14,2)=0: RANGE(14,3)=1: RANGE(14,4)=0
1780 RANGE(15,1)=0: RANGE(15,2)=1: RANGE(15,3)=0: RANGE(15,4)=0
1790 RANGE(16,1)=0: RANGE(16,2)=0: RANGE(16,3)=2/3: RANGE(16,4)=1/3
1800 RANGE(17,1)=2/3: RANGE(17,2)=1/3: RANGE(17,3)=0: RANGE(17,4)=0
1810 RANGE(18,1)=1: RANGE(18,2)=0: RANGE(18,3)=0: RANGE(18,4)=0
1820 RANGE(19,1)=2/3: RANGE(19,2)=0: RANGE(19,3)=1/3: RANGE(19,4)=0
1830 RANGE(20,1)=2/3: RANGE(20,2)=0: RANGE(20,3)=0: RANGE(20,4)=1/3
1840 '
1850 '                number for each box type of a pallet pattern
1860 '
1870 NTYPE(1,1)=0 : NTYPE(1,2)=4 : NTYPE(1,3)=4 : NTYPE(1,4)=4
1880 NTYPE(2,1)=0 : NTYPE(2,2)=4 : NTYPE(2,3)=7 : NTYPE(2,4)=0
1890 NTYPE(3,1)=4 : NTYPE(3,2)=0 : NTYPE(3,3)=4 : NTYPE(3,4)=4
1900 NTYPE(4,1)=0 : NTYPE(4,2)=10 : NTYPE(4,3)=5 : NTYPE(4,4)=0
1910 NTYPE(5,1)=5 : NTYPE(5,2)=5 : NTYPE(5,3)=5 : NTYPE(5,4)=0
1920 NTYPE(6,1)=0 : NTYPE(6,2)=12 : NTYPE(6,3)=0 : NTYPE(6,4)=6
1930 NTYPE(7,1)=4 : NTYPE(7,2)=0 : NTYPE(7,3)=7 : NTYPE(7,4)=0
1940 NTYPE(8,1)=7 : NTYPE(8,2)=7 : NTYPE(8,3)=0 : NTYPE(8,4)=7
1950 NTYPE(9,1)=0 : NTYPE(9,2)=4 : NTYPE(9,3)=0 : NTYPE(9,4)=8
1960 NTYPE(10,1)=0: NTYPE(10,2)=0: NTYPE(10,3)=2: NTYPE(10,4)=4
1970 NTYPE(11,1)=12: NTYPE(11,2)=25: NTYPE(11,3)=0: NTYPE(11,4)=0
1980 NTYPE(12,1)=0: NTYPE(12,2)=0: NTYPE(12,3)=0: NTYPE(12,4)=8
1990 NTYPE(13,1)=4: NTYPE(13,2)=0: NTYPE(13,3)=0: NTYPE(13,4)=8
2000 NTYPE(14,1)=0: NTYPE(14,2)=0: NTYPE(14,3)=8: NTYPE(14,4)=0
```

```
2010 NTYPE(15,1)=0: NTYPE(15,2)=32: NTYPE(15,3)=0: NTYPE(15,4)=0
2020 NTYPE(16,1)=0: NTYPE(16,2)=0: NTYPE(16,3)=4: NTYPE(16,4)=2
2030 NTYPE(17,1)=32: NTYPE(17,2)=16: NTYPE(17,3)=0: NTYPE(17,4)=0
2040 NTYPE(18,1)=64: NTYPE(18,2)=0: NTYPE(18,3)=0: NTYPE(18,4)=0
2050 NTYPE(19,1)=12: NTYPE(19,2)=0: NTYPE(19,3)=6: NTYPE(19,4)=0
2060 NTYPE(20,1)=16: NTYPE(20,2)=0: NTYPE(20,3)=0: NTYPE(20,4)=8
2070 '
2080 PR=1 : F1$="PLT" : F3$=".DAT"
2090 PV=64 : BV(1)=1 : BV(2)=2 : BV(3)=8 : BV(4)=6
2100 GOSUB 6430 : OLDTM1=DNEWTM : STM1=DNEWTM : NBOX=0
2110 OLDTM2=DNEWTM : STM2=DNEWTM : GOSUB 6530
2120 FILEN=DSEQ(1) : GOSUB 6290
2130 '
2140 '          read data of a pallet pattern
2150 '
2160 CLOSE#3 : OPEN PATTERN$ FOR INPUT AS #3
2170 ANGL=360/TNP
2180 INPUT #3,TYPE,NODE,ACT
2190 FOR I=1 TO NODE
2200 INPUT #3,TE,TYPE(I),PX(I),PY(I),PZ(I),PO(I),MT6A(I),MT6B(I)
2210 NEXT I
2220 FOR I=1 TO NODE : PNT(I)=0 : NEXT I
2230 FOR I=1 TO ACT : INPUT #3,T(I),B(I) : NEXT I
2240 FOR I=1 TO 5 : FOR J=1 TO 5 : START(I,J)=0 : NEXT J : NEXT I
2250 MAX=NODE : IF MAX < ACT THEN MAX=ACT
2260 '
2270 '          construct chains
2280 '
2290 FOR I=1 TO TNP+1 : FOR J=1 TO MAX
2300 NOPRE(I,J)=0 : INDEX(I,J)=0 : INVR(I,J)=0
2310 NEXT J : NEXT I
2320 B(ACT+1)=0 : T(ACT+1)=0 : J=1 : NO=T(1)
2330 NOPRE(1,B(1))=NOPRE(1,B(1))+1
2340 FOR I=2 TO ACT+1
2350 NOPRE(1,B(I))=NOPRE(1,B(I))+1
2360 IF NO=T(I) GOTO 2380
2370 PNT(NO)=J : NO=T(I) : J=I
2380 NEXT I
2390 FOR I=1 TO TYPE : T(I)=0 : NEXT I
2400 FOR I=1 TO NODE
2410 IF T(TYPE(I))=0 THEN START(1,TYPE(I))=I
2420 INDEX(1,T(TYPE(I)))=I : T(TYPE(I))=I
2430 NEXT I
2440 FOR I=1 TO TYPE : T(I)=0 : NEXT I
2450 FOR I=NODE TO 1 STEP -1
2460 INVR(1,T(TYPE(I)))=I : T(TYPE(I))=I
2470 NEXT I  :  INDEX(1,0)=0
2480 FOR I=2 TO TNP+1 : FOR J=0 TO MAX
2490 NOPRE(I,J)=NOPRE(1,J) : INDEX(I,J)=INDEX(1,J)
2500 INVR(I,J)=INVR(1,J)
```

```
2510 NEXT J : NEXT I
2520 FOR I=2 TO TNP+1 : FOR J=1 TO TYPE
2530 START(I,J)=START(1,J)
2540 NEXT J : NEXT I
2550 FOR I=1 TO TNP : PALLET(I)=NODE : NEXT I
2560 GOSUB 6430 : CYCTM=DNEWTM
2565 '
2570 '         pick up a box from in-feeding conveyor
2575 '
2580 GOSUB 6430 : CCT1=CCT1+DNEWTM-CYCTM
2590 CCT2=CCT2+DNEWTM-CYCTM : CYCTM=DNEWTM
2600 OPT=DNEWTM : INDXP=0
2610 NBOX=NBOX+1 : IF NBOX<=QO THEN BT=NBSEQ(1,NBOX) : GOTO 2720
2620 GOSUB 6740
2630 GOSUB 6430 : OLDTM1=DNEWTM
2640 STM1=DNEWTM : CCT1=0 : NCLD1=0 : OPTS1=0
2650 FOR J=1 TO 4
2660 NMAX1(J)=NIQ(J) : CTM1(J)=0 : NTTLE1(J)=0 : NGES1(J)=0
2670 DCTM(J)=0 : DNMAX(J)=0 : DNIQ(J)=0
2680 NEXT J
2690 NDIST=NDIST+1 : IF NDIST > NPTTN THEN GOSUB 7010
2700 NBOX=0 : IDK=1 : GOSUB 6530
2710 GOSUB 7340
2720 NGES1(BT)=NGEST1(BT)+1 : NGES2(BT)=NGES2(BT)+1
2730 FOR J=1 TO 5 : C(J)=P2(BT,J)-P1(J) : NEXT J
2740 C(5)=ABS(C(5)) : GOSUB 5120
2750 'PRINT #1,"C+09" : ' actuate the gripper
2760 GOSUB 6380
2770 FOR J=1 TO 4 : C(J)=-C(J) : NEXT J : GOSUB 5120
2780 DT=MT2 : GOSUB 7270
2790 '
2800 '         determine where to place the box
2810 '         NC=0 for pallet; NC=1 for storage
2820 '
2830 GOSUB 4010
2840 IF NC=0 GOTO 3910
2850 IF NC=1 GOTO 2880
2860 PRINT "ERROR" : STOP
2870 '
2880 '         place the box in storage area
2890 '
2900 ID$="S" : SAN=BT
2910 X=SA(BT,1) : Y=SA(BT,2) : Z=HZ : GOSUB 4890
2920 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
2930 FOR J=1 TO 5 : C(J)=PU(J)-P1(J) : NEXT J
2940 C(5)=ABS(C(5)) : GOSUB 5120
2950 X=SA(BT,1) : Y=SA(BT,2) : Z=SA(BT,3)+AF : GOSUB 4890
2960 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
2970 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
2980 C(5)=ABS(C(5)) : GOSUB 5120
```

```
2990 FOR J=1 TO 5 : T(J)=PD(J) : NEXT J
3000 TX=SA(BT,1) : TY=SA(BT,2) : TZ=SA(BT,3) : GOSUB 5750
3010 'PRINT #1,"CX"
3020 GOSUB 6380 : ' release the gripper
3030 GOSUB 6430 : CTM1(BT)=CTM1(BT)+(DNEWTM-OLDTM1)*NIQ(BT)
3040 CTM2(BT)=CTM2(BT)+(DNEWTM-OLDTM2)*NIQ(BT)
3050 DCTM(BT)=DCTM(BT)+(DNEWTM-OLDTM1)*DNIQ(BT)
3060 NIQ(BT)=NIQ(BT)+1 : DNIQ(BT)=DNIQ(BT)+1
3070 OLDTM1=DNEWTM : OLDTM2=DNEWTM
3080 GOSUB 5910 : ' straight up
3090 IF NMAX1(BT) < NIQ(BT) THEN NMAX1(BT)=NIQ(BT)
3100 IF DNMAX(BT) < DNIQ(BT) THEN DNMAX(BT)=DNIQ(BT)
3110 IF NMAX2(BT) < NIQ(BT) THEN NMAX2(BT)=NIQ(BT)
3120 NTTLE1(BT)=NTTLE1(BT)+1 : NTTLE2(BT)=NTTLE2(BT)+1
3130 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
3140 C(5)=ABS(C(5)) : GOSUB 5120
3150 DT=MT3(BT) : GOSUB 7270
3160 '
3170 '                  update placement location in storage area
3180 '
3190 SA(BT,1)=SA(BT,1)+WH(BT)
3200 IF SA(BT,1) <= LX(BT) GOTO 3380
3210 SA(BT,1)=IX(BT) : SA(BT,2)=SA(BT,2)+LH(BT)
3220 IF SA(BT,2) <= LY(BT) GOTO 3380
3230 SA(BT,2)=IY(BT) : SA(BT,3)=SA(BT,3)+HT(BT)
3240 IF SA(BT,3) <= LZ(BT) GOTO 3380
3250 '
3260 '                 storage overflow
3270 '                 (infinite capacity is assumed)
3280 '
3290 'PRINT "STORAGE OVERFLOW -- CLEAN THE STORAGE"
3300 'FOR J=1 TO 3 : PLAY "C+L2" : NEXT J
3310 'PRINT : PRINT "ENTER ANY KEY TO CONTINUE " : INPUT KY$ : CLS
3320 SA(BT,1)=IX(BT) : SA(BT,2)=IY(BT) : SA(BT,3)=IZ(BT)
3330 STOVFL(BT)=STOVFL(BT)+1
3340 'FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
3350 'C(5)=ABS(C(5)) : GOSUB 5120
3360 'GOTO 2570 : ' go to pick-up position
3370 '
3380 '          mover one box from every storage
3390 '          and place it onto the pallet
3400 '
3410 PR1=0
3420 ICUM=0
3430 FOR N=1 TO TYPE
3440 IF RANGE(FILEN,N)=0 THEN ICUM=ICUM+1 : GOTO 3800
3450 IF NIQ(N)=0 THEN ICUM=ICUM+1 : GOTO 3800
3460 BT=N : GOSUB 4010
3470 IF NC=1 THEN ICUM=ICUM+1 : GOTO 3800 : ' no pallet space avail.
3480 SA(N,1)=SA(N,1)-WH(N)
```

```
3490 IF SA(N,1) >= IX(N) GOTO 3550
3500 SA(N,1)=LX(N) : SA(N,2)=SA(N,2)-LH(N)
3510 IF SA(N,2) >= IY(N) GOTO 3550
3520 SA(N,2)=LY(N) : SA(N,3)=SA(N,3)-HT(N)
3530 IF SA(N,3) >= IZ(N) GOTO 3550
3540 SA(N,1)=IX(N) : SA(N,2)=IY(N) : SA(N,3)=IZ(N)
3550 X=SA(N,1) : Y=SA(N,2) : Z=HZ : GOSUB 4890
3560 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
3570 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
3580 C(5)=ABS(C(5)) : GOSUB 5120
3590 X=SA(N,1) : Y=SA(N,2) : Z=SA(N,3)+AF : GOSUB 4890
3600 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
3610 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
3620 C(5)=ABS(C(5)) : GOSUB 5120
3630 FOR J=1 TO 5 : T(J)=PU(J) : NEXT J
3640 TX=SA(N,1) : TY=SA(N,2) : TZ=SA(N,3) : GOSUB 5750
3650 'PRINT #1,"C+9"
3660 GOSUB 6380 : GOSUB 5910
3670 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
3680 C(5)=ABS(C(5)) : GOSUB 5120
3690 IF ID$="S" THEN DT=MT4A(SAN,BT) ELSE DT=MT4B(BT)
3700 GOSUB 7270
3710 GOSUB 6430
3720 CTM1(BT)=CTM1(BT)+(DNEWTM-OLDTM1)*NIQ(BT)
3730 CTM2(BT)=CTM2(BT)+(DNEWTM-OLDTM2)*NIQ(BT)
3740 DCTM(BT)=DCTM(BT)+(DNEWTM-OLDTM1)*DNIQ(BT)
3750 NIQ(BT)=NIQ(BT)-1 : DNIQ(BT)=DNIQ(BT)-1
3760 IF DNIQ(BT) < 0 THEN DNIQ(BT)=0
3770 OLDTM1=DNEWTM : OLDTM2=DNEWTM : INDXP=0
3780 IDX$="SA" : SAN=BT
3790 GOSUB 4340 : ' place box onto the pallet
3800 NEXT N
3810 IF ICUM <> TYPE AND PALLET(PR)/NODE <= PRCTG THEN
                          PR1=1 : GOTO 3420
3830 GOSUB 6430 : OPT=DNEWTM
3840 FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
3850 C(5)=ABS(C(5)) : GOSUB 5120
3860 IF ID$="S" THEN DT=MT5A(SAN) ELSE DT=MT5B
3870 GOSUB 7270 : GOSUB 6430
3880 OPTS1=OPTS1+DNEWTM-OPT : OPTS2=OPTS2+DNEWTM-OPT
3890 GOTO 2570 : ' go to pick-up position
3900 '
3910 '         the pallet
3920 '
3930 INDXP=1 : IDX$="P1"
3940 FOR J=1 TO 5 : PD(J)=P1(J) : NEXT J
3950 GOSUB 4340 : ' place box onto pallet
3960 GOSUB 6430
3970 OPTS1=OPTS1+DNEWTM-OPT : OPTS2=OPTS2+DNEWTM-OPT : INDXP=0
3980 GOTO 3380 : ' move one box from every storage area
```

```
3990 END
4000 '
4010 '         search the chain for box's placement location
4020 '
4030 T(1)=PR : TE=PR : NC=0
4040 IF TNP=1 GOTO 4080
4050 FOR I=2 TO TNP
4060 TE=TE+1 : IF TE > TNP THEN TE=1
4070 T(I)=TE : NEXT I
4080 FOR K=1 TO TNP : TE=T(K) : SEQ=K
4090 SRCH=START(TE,BT)
4100 FOR I=1 TO NODE
4110 IF START(TE,BT)=0 GOTO 4170
4120 IF NOPRE(TE,SRCH)=0 THEN GOTO 4200
4130 SRCH=INDEX(TE,SRCH)
4140 IF SRCH=0 GOTO 4170
4150 NEXT I
4160 IF PR1=1 GOTO 4180
4170 NEXT K
4180 NC=1 : RETURN : ' no pallet space available
4190 '
4200 '                  update the chain
4210 '
4220 IF START(TE,BT)=SRCH THEN
              START(TE,BT)=INDEX(TE,SRCH) : GOTO 4260
4240 INDEX(TE,INVR(TE,SRCH))=INDEX(TE,SRCH)
4250 INVR(TE,INDEX(TE,SRCH))=INVR(TE,SRCH)
4260 IF PNT(SRCH)=0 GOTO 4320
4270 NXT=SRCH : FOR I=1 TO NODE : NXT=NXT+1
4280 IF NXT=NODE+1 THEN NO=ACT : GOTO 4310
4290 IF PNT(NXT) <> 0 THEN NO=PNT(NXT)-1 : GOTO 4310
4300 NEXT I : PRINT "ERROR ON PNT" : STOP
4305 FOR K=PNT(SRCH) TO NO
4310 NOPRE(TE,B(K))=NOPRE(TE,B(K))-1
4315 NEXT K
4320 PLTN=TE : NC=0 : RETURN
4330 '
4340 '         place a box onto the pallet
4350 '
4360 IF ANGL*(SEQ-1) <= 180 GOTO 4380
4370 SPN$(1,1)="-" : RTH=(360/ANGL-SEQ+1)*2620/TNP : GOTO 4390
4380 SPN$(1,1)="+" : RTH=2620/TNP*(SEQ-1)
4390 GOSUB 5990 : ' rotate table for desired pallet
4400 X=PX(SRCH) : Y=PY(SRCH) : Z=HZ : GOSUB 4890
4410 PU(1)=T1 : PU(2)=T2 : PU(3)=T3 : PU(4)=T4 : PU(5)=ANGLE
4420 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
4430 C(5)=ABS(C(5)) : GOSUB 5120
4440 IF PO(SRCH) <> 1 GOTO 4460
4450 SPN$(2,2)="+" : GOSUB 6140
4460 X=PX(SRCH) : Y=PY(SRCH) : Z=PZ(SRCH)+TABLEH+AF : GOSUB 4890
```

```
4470 PD(1)=T1 : PD(2)=T2 : PD(3)=T3 : PD(4)=T4 : PD(5)=ANGLE
4480 FOR J=1 TO 5 : C(J)=PD(J)-PU(J) : NEXT J
4490 C(5)=ABS(C(5)) : GOSUB 5120
4500 FOR J=1 TO 5 : T(J)=PD(J) : NEXT J
4510 TX=PX(SRCH) : TY=PY(SRCH) : TZ=PZ(SRCH)+TABLEH : GOSUB 5750
4520 PRINT #1,"CX" : GOSUB 6380 : GOSUB 5910
4530 FOR J=1 TO 5 : C(J)=PU(J)-PD(J) : NEXT J
4540 C(5)=ABS(C(5)) : GOSUB 5120
4550 IF IDX$="SA" THEN DT=MT6B(SRCH) ELSE DT=MT6A(SRCH)
4560 GOSUB 7270
4570 IF SPN$(1,1)="+" THEN SPN$(1,1)="-" ELSE SPN$(1,1)="+"
4580 GOSUB 5990 : ' rotate table to original pallet
4590 PALLET(PLTN)=PALLET(PLTN)-1 : NCLD1=NCLD1+1 : NCLD2=NCLD2+1
4600 IF PO(SRCH) <> 1 GOTO 4620
4610 SPN$(2,2)="-" : GOSUB 6140
4620 IF PALLET(PLTN) <> 0 THEN RETURN
4630 '
4640 '                pallet is full
4650 '                (change pallet pattern according to
4660 '                in-coming box distribution)
4670 '
4680 FOR J=1 TO 5 : C(J)=P1(J)-PU(J) : NEXT J
4690 C(5)=ABS(C(5)) : GOSUB 5120
4700 DT=MT5B : GOSUB 7270
4710 IF PR=TNP THEN PR=1 ELSE PR=PR+1
4720 NO=TNP+1
4730 FOR J=0 TO MAX
4740 NOPRE(PLTN,J)=NOPRE(NO,J) : INDEX(PLTN,J)=INDEX(NO,J)
4750 INVR(PLTN,J)=INVR(NO,J)
4760 NEXT J
4770 FOR J=1 TO TYPE : START(PLTN,J)=START(NO,J) : NEXT J
4780 PALLET(PLTN)=NODE
4790 RTH=655
4800 SPN$(1,1)="+" : GOSUB 5990 : ' rotate the turntable
4810 'PRINT "REMOVE THE PALLET AND INSERT A NEW PALLET" : PRINT
4820 'PRINT "ENTER ANY KEY WHEN READY" : INPUT SPP$(1,1) : CLS
4830 IF INDXP=0 GOTO 4860
4840 GOSUB 6430
4850 OPTS1=OPTS1+DNEWTM-OPT : OPTS2=OPTS2+DNEWTM-OPT : INDXP=0
4860 GOSUB 7340 : ' use dynamic selection proc. for a pattern
4870 RETURN
4880 '
4890 '        coordinate transformation
4900 '
4910 RR=SQR(X*X+Y*Y)
4920 IF X = 0 THEN T1=SGN(Y)*PI/2
4930 IF X > 0 THEN T1=ATN(Y/X)
4940 IF X < 0 AND Y > 0 THEN T1=PI-ATN(Y/ABS(X))
4950 IF X < 0 AND Y < 0 THEN T1=-(PI-ATN(Y/X))
4960 ANGLE=ABS(T1*C) : IF X=0 THEN T4=0 ELSE T4=ATN(ABS(Y/X))
```

```
4970 RO=RR-LL*COS(P)
4980 ZO=Z-LL*SIN(P)-H
4990 IF RO=0 THEN G=SGN(ZO)*P!/2! ELSE G=ATN(ZO/RO)
5000 A=RO*RO+ZO*ZO
5010 A=4*L*L/A-1
5020 A=ABS(A)
5030 A=ATN(SQR(A))
5040 T2=A+G
5050 T3=G-A
5060 T1=INT(T1*SF)
5070 T2=INT(T2*SE)
5080 T3=INT(T3*SD)
5090 T4=INT(T4*SA)
5100 RETURN
5110 '
5120 '         simultaneous movements of joints
5130 '
5140 RETURN : ' The subroutine is skipped in this simulation
5150 SIGN$(1)="+" : SIGN$(2)="+" : SIGN$(3)="+" : SIGN$(4)="+"
5160 IF C(1) < 0 THEN SIGN$(1)="-"
5170 IF C(2) > 0 THEN SIGN$(2)="-"
5180 IF C(3) < 0 THEN SIGN$(3)="-"
5190 IF C(5) <= 90 AND C(1) < 0 THEN SIGN$(4)="-"
5200 IF C(5) <= 90 AND C(1) > 0 THEN SIGN$(4)="+"
5210 IF C(5) > 90 AND C(1) < 0 THEN SIGN$(4)="+"
5220 IF C(5) > 90 AND C(1) > 0 THEN SIGN$(4)="-"
5230 '
5240 HO(1)=ABS(C(1)) : HO(2)=ABS(C(2)) : HO(3)=ABS(C(3))
5250 FOR J1=1 TO 2 : L1=J1 : JJ=J1+1 : FOR J2=JJ TO 3
5260 IF HO(L1) < HO(J2) THEN L1=J2
5270 NEXT J2
5280 TE=HO(J1) : HO(J1)=HO(L1) : HO(L1)=TE
5290 TE=IDX(J1) : IDX(J1)=IDX(L1) : IDX(L1)=TE
5300 NEXT J1
5310 FOR I=1 TO 3
5320 MO$(I)=MTR$(IDX(I)) : SO$(I)=SIGN$(IDX(I)) : NEXT I
5330 NEXT I
5340 '
5350 IF HO(1)=0 GOTO 5600
5360 E1=0 : COUNT=0 : RATE1=HO(2)/HO(1) : RATE2=HO(3)/HO(1)
5370 D1=0 : HS=HO(1)
5380 PRINT #1,MO$(1);"?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5390 IF CD > Z2 GOTO 5380
5400 IF HS > Z1 GOTO 5430
5410 TN$=CHR$(HS+48) : PRINT #1,MO$(1);SO$(1);TN$
5420 COUNT=COUNT+HS : GOTO 5450
5430 PRINT #1,MO$(1);SO$(1);ZO$
5440 COUNT=COUNT+Z1
5450 HS=HS-Z1
5460 IF HO(2)=0 GOTO 5560
```

```
5470 E2=INT(COUNT*RATE1) : TE=E2-E1
5480 IF TE=0 GOTO 5510
5490 E1=E2 : TN$=CHR$(TE+48)
5500 PRINT #1,MO$(2);SO$(2);TN$
5510 IF HO(3)=0 GOTO 5560
5520 D2=INT(COUNT*RATE2) : TE=D2-D1
5530 IF TE=0 GOTO 5560
5540 D1=D2 : TN$=CHR$(TE+48)
5550 PRINT #1,MO$(3);SO$(3);TN$
5560 IF HS > 0 GOTO 5380
5570 '
5580 '                  roll the gripper
5590 '
5600 HS=ABS(C(4)) : IF HS=0 GOTO 5680
5610 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
5620 IF CD > Z2 GOTO 5610
5630 IF HS > Z1 GOTO 5650
5640 TN$=CHR$(HS+48) : PRINT #1,MTR$(4);SIGN$(4);TN$ : GOTO 5680
5650 PRINT #1,MTR$(4);SIGN$(4);ZO$
5660 HS=HS-Z1
5670 IF HS > 0 GOTO 5610
5680 CD=0
5690 FOR I=1 TO 4 : PRINT #1,MTR$(I);"?"
5700 TN$=INPUT$(1,#1) : CD=CD+ASC(TN$)-32
5710 NEXT I
5720 IF CD <> 0 GOTO 5680
5730 RETURN
5740 '
5750 '         move straight down
5760 '
5770 JM=AF/NN : SZ=TZ+AF
5780 FOR J=1 TO NN
5790 X=TX : Y=TY : Z=SZ-JM*J : GOSUB 4890
5800 TYPE(1)=T1 : TYPE(2)=T2 : TYPE(3)=T3
5810 TYPE(4)=T4 : TYPE(5)=ANGLE
5820 FOR JP=1 TO 5 : JK(J,JP)=TYPE(JP)-T(JP) : NEXT JP
5830 JK(J,5)=ABS(JK(J,5))
5840 FOR JP=1 TO 5 : T(JP)=TYPE(JP) : NEXT JP
5850 NEXT J
5860 FOR J=1 TO NN
5870 FOR JP=1 TO 5 : C(JP)=JK(J,JP) : NEXT JP : GOSUB 5120
5880 NEXT J
5890 RETURN
5900 '
5910 '         move straight up
5920 '
5930 FOR J=NN TO 1 STEP -1
5940 FOR JP=1 TO 4 : C(JP)=-JK(J,JP) : NEXT JP
5950 C(5)=ABS(JK(J,5)) : GOSUB 5120
5960 NEXT J
```

```
5970 RETURN
5980 '
5990 '        rotate the turntable
6000 '
6010 DT=MT7 : GOSUB 7270
6020 RETURN : ' this subroutine is skipped in this simulation
6030 HS=RTH
6040 PRINT #1,"H?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6050 IF CD > 85 GOTO 6040
6060 IF HS >= 10 GOTO 6080
6070 TN$=CHR$(HS+48) : PRINT #1,"H";SPN$(1,1);TN$ : GOTO 6100
6080 PRINT #1,"H";SPN$(1,1);"10" : HS=HS-10
6090 IF HS > 0 GOTO 6040
6100 PRINT #1,"H?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6110 IF CD <> 0 GOTO 6100
6120 RETURN
6130 '
6140 '        roll the gripper for changing box orientation
6150 '
6160 DT=MT8 : GOSUB 7270
6170 RETURN : ' this subroutine is skipped in this simulation
6180 HS=375
6190 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6200 IF CD > 85 GOTO 6190
6210 IF HS >= 6 GOTO 6230
6220 TN$=CHR$(HS+48) : PRINT #1,"A";SPN$(2,2);TN$ : GOTO 6250
6230 PRINT #1,"A";SPN$(2,2);"6" : HS=HS-6
6240 IF HS > 0 GOTO 6190
6250 PRINT #1,"A?" : A$=INPUT$(1,#1) : CD=ASC(A$)-32
6260 IF CD <> 0 GOTO 6250
6270 RETURN
6280 '
6290 '        convert number to characters for data file name
6300 '
6310 IF FILEN < 10 THEN F2$=CHR$(FILEN+48) : GOTO 6340
6320 E1=INT(FILEN/10) : R10$=CHR$(E1+48)
6330 E2=FILEN-E1*10 : R01$=CHR$(E2+48) : F2$=R10$+R01$
6340 PATTERN$=F1$+F2$+F3$
6350 CPP=CPP+1 : PRINT#5,"PP ";CPP;" : ";PATTERN$
6360 RETURN
6370 '
6380 '        delay for picking up or placing a box
6390 '
6400 FOR J=1 TO 500 : E1=SQR(2) : NEXT J
6410 RETURN
6420 '
6430 '        obtain current time
6440 '
6450 SNAP1$=DATE$ : SNAP2$=TIME$
6460 DNEWTM=VAL(MID$(SNAP1$,4,2))*86400!+VAL(LEFT$(SNAP2$,2))*3600!
```

```
6470 DNEWTM=DNEWTM+VAL(MID$(SNAP2$,4,2))*60!
6480 DNEWTM=DNEWTM+VAL(RIGHT$(SNAP2$,2))-86400!
6490 RETURN
6500 '
6510 '        generate box types (two distributions at a time)
6520 '
6530 FOR J=1 TO QO : NBSEQ(1,J)=NBSEQ(2,J) : NEXT J
6540 STRT=2 : IF NDIST=1 THEN STRT=1
6550 IF NDIST=NPTTN GOTO 6700
6560 FOR K=STRT TO 2 : FILEN=DSEQ(NDIST+K-1)
6570 E1=0
6580 FOR J=1 TO TYPE
6590 E1=E1+RANGE(FILEN,J) : PROB(J)=E1*QO
6600 NEXT J
6610 FOR J=1 TO QO : NSEQ(J)=J : NEXT J
6620 FOR Q1=QO TO 1 STEP -1
6630 E1=INT(RND*Q1)+1
6640 FOR J=1 TO TYPE
6650 IF NSEQ(E1)<=PROB(J) THEN BT=J : GOTO 6680
6660 NEXT J
6670 PRINT "ERROR ON BT" : STOP
6680 NBSEQ(K,QO-Q1+1)=BT : NSEQ(E1)=NSEQ(Q1)
6690 NEXT Q1 : NEXT K
6700 RETURN
6710 '
6720 '        deliver statistics of a distribution run
6730 '
6740 GOSUB 6430 : PRTTM=DNEWTM
6750 FOR J=1 TO 4 : CTM1(J)=CTM1(J)+(DNEWTM-OLDTM1)*NIQ(J)
6760 DCTM(J)=DCTM(J)+(DNEWTM-OLDTM1)*DNIQ(J) : NEXT J
6770 PRINT#4,"DIST. NUMBER: ";NDIST
6780 PRINT#4,"TOTAL NUMBER OF CARTONS LOADED TO THE PALLET: ";NCLD1
6790 PRINT#4,"TOTAL SIMULATION TIME (DIST): ";DNEWTM-STM1
6800 PRINT#4,"AVERAGE CYCLE TIME (PICK-UP): ";CCT1/QO
6810 PRINT#4,"OPERATION TIME IN STORAGE AREA: ";DNEWTM-STM1-OPTS1
6820 FOR J=1 TO 4
6830 PRINT#4,"CUMULATIVE TIME AREA: ";J,CTM1(J),DCTM(J)
6840 AVE1=CTM1(J)/(DNEWTM-STM1) : AVE2=DCTM(J)/(DNEWTM-STM1)
6850 PRINT#4,"AVERAGE CONTENTS: ";J,AVE1,AVE2
6860 IF NTTLE1(J)=0 THEN
          PRINT#4,"AVERAGE WAITING TIME: ";J,0,0 : GOTO 6900
6880 AVE1=CTM1(J)/NTTLE1(J) : AVE2=DCTM(J)/NTTLE1(J)
6890 PRINT#4,"AVERAGE WAITING TIME: ";J,AVE1,AVE2
6900 PRINT#4,"TOTAL CARTONS ENTERED: ";J,NGES1(J)
6910 PRINT#4,"MAXIMUM CONTENTS: ";J,NMAX1(J),DNMAX(J)
6920 PRINT#4,"TOTAL ENTRIES: ";J,NTTLE1(J)
6930 PRINT#4,"ZERO ENTRIES: ";J,NGES1(J)-NTTLE1(J)
6940 PRINT#4,"CURRENT CONTENTS: ";J,NIQ(J),DNIQ(J)
6950 PRINT#4,"PROPORTIONS: ";J, RANGE(NDIST,J)
6960 NEXT J
```

```
6970 GOSUB 6430
6980 IF NDIST <> NPTTN THEN CPRTM=CPRTM+(DNEWTM-PRTTM)
6990 RETURN
7000 '
7010 '        deliver statistics of total simulation
7020 '
7030 FOR J=1 TO 4
7040 CTM2(J)=CTM2(J)+(PRTTM-OLDTM2)*NIQ(J)-CPRTM
7050 NEXT J
7060 PRINT#4,">>>SYSTEM SIMULATION STATISTICS"
7070 PRINT#4,"TOTAL NUMBER OF CARTONS LOADED TO THE PALLET: ";NCLD2
7080 PRINT#4,"TOTAL SIMULATION TIME (ENTIRE): ";PRTTM-STM2
7090 PRINT#4,"AVERAGE CYCLE TIME (PICK-UP): ";CCT2/(QO*NPTTN)
7100 PRINT#4,"OPERATION TIME IN STORAGE AREA: ";PRTTM-STM2-OPTS2
7110 FOR J=1 TO 4
7120 PRINT#4,"CUMULATIVE TIME AREA: ";J,CTM2(J)
7130 PRINT#4,"AVERAGE CONTENTS: ";J,CTM2(J)/(PRTTM-STM2)
7140 IF NTTLE2(J)=0 THEN
        PRINT #4,"AVERAGE WAITING TIME: ";J,0,0 : GOTO 7170
7160 PRINT#4,"AVERAGE WAITING TIME: ";J,CTM2(J)/NTTLE2(J)
7170 PRINT#4,"TOTAL CARTONS ENTERED: ";J,NGES2(J)
7180 PRINT#4,"MAXIMUM CONTENTS: ";J,NMAX2(J)
7190 PRINT#4,"TOTAL ENTRIES: ";J,NTTLE2(J)
7200 PRINT#4,"ZERO ENTRIES: ";J,NGES2(J)-NTTLE2(J)
7210 PRINT#4,"NUMBER OF OVERFLOWS: ";J,STOVFL(J)
7220 PRINT#4,"CURRENT CONTENTS: ";J,NIQ(J)
7230 NEXT J
7240 CLS : PRINT "READ OUTPUT.DAT"
7250 END : RETURN
7260 '
7270 '        delay for robot movement time
7280 '
7290 GOSUB 6430 : DT=DNEWTM+DT
7300 GOSUB 6430 : TE=DNEWTM
7310 IF TE < DT GOTO 7300
7320 RETURN
7330 '
7340 '        dynamic selection for a best matched pallet pattern
7350 '        FCT = the look-ahead factor
7360 '        BRATIO(i) = box ratio of pallet pattern i
7370 '
7380 TVOL=0 : IDK=1 : FOR J=1 TO TYPE : QTYPE(J)=0 : NEXT J
7390 TNBOX=NBOX
7400 TNBOX=TNBOX+1
7410 IF TNBOX > QO AND NDIST=NPTTN GOTO 7460
7420 IF TNBOX > QO THEN IDK=2 : TNBOX=1
7430 QTYPE(NBSEQ(IDK,TNBOX))=QTYPE(NBSEQ(IDK,TNBOX))+1
7440 TVOL=TVOL+BV(NBSEQ(IDK,TNBOX))
7450 IF TVOL < FCT*PV GOTO 7400
7460 TBOX=0 : FOR J=1 TO TYPE : TBOX=TBOX+QTYPE(J) : NEXT J
```

```
7470 IF TBOX=0 THEN TBOX=1
7480 FOR I=1 TO NPTTN : BRATIO(I)=0
7490 FOR J=1 TO TYPE
7500 BRATIO(I)=BRATIO(I)+ABS(QTYPE(J)/TBOX-RANGE(I,J)) : NEXT J
7510 FOR J=1 TO TYPE
7520 IF QTYPE(J)=0 AND NTYPE(I,J)>0 THEN CHECK(I)=2 : GOTO 7560
7530 IF (QTYPE(J)>0 AND NTYPE(I,J)=0) OR QTYPE(J)<NTYPE(I,J) THEN
                        CHECK(I)=0 : GOTO 7560
7550 NEXT J : CHECK(I)=1
7560 NEXT I : MINX1=100 : MINX2=100 : MINX3=100
7570 FOR I=1 TO NPTTN
7580 IF CHECK(I)=1 AND MINX1 > BRATIO(I) THEN
                        MINX1=BRATIO(I) : FILEN1=I
7600 IF CHECK(I)=0 AND MINX2 > BRATIO(I) THEN
                        MINX2=BRATIO(I) : FILEN2=I
7620 IF MINX3 > BRATIO(I) THEN MINX3=BRATIO(I) : FILEN3=I
7630 NEXT I
7640 IF MINX1 <> 100 THEN FILEN=FILEN1 : GOTO 7670
7650 IF MINX2 <> 100 THEN FILEN=FILEN2 : GOTO 7670
7660 FILEN=FILEN3 : GOTO 7680
7670 IF (MINX2 < MINX1) AND (MINX2 < .1) THEN FILEN=FILEN2
7680 GOSUB 6290 : GOTO 2140
7690 RETURN
```

XIV. APPENDIX F.

DETERMINATION OF PALLET PATTERNS

The heuristic dynamic programming model consists of two goals. The first is to maximize the utilization of a pallet cube. The second is to make the proportions of boxes satisfy some user-specified numbers. Because of the design nature of the heuristic dynamic programming model, the procedure tends to maximize the pallet space occupied by boxes, and sacrifice the desired box proportions. In the simulation, the box proportions of a pallet pattern have been made to be as close as possible to the user-specified numbers so that the performance and feasibility of a robotic palletizing system can be evaluated. Post-solution adjustment is required to obtain the desired box proportions. Table F-1 lists the number of boxes used in each pallet pattern. In Table F-1, each pallet pattern consists of four data categories A, B, C and D. Category A is the input number for each type of box to the heuristic DP program. These input numbers are determined based on eq. (3.1). In the event that the numbers determined from eq. (3.1) do not generate a rational solution, the input number of the box size that has largest volume is increased by 1. The input number of the box size which has smallest volume is then decreased to an amount such that the total box volumes of all sizes do not exceed the pallet's volume. The strategy of rounding up or increasing the number of larger boxes is that the larger boxes can be always replaced by smaller boxes. The feasibility of the solution holds with this substitution. The post-solution adjustment can then be carried out to yield desired box proportions.

Category B gives the solution of the heuristic DP program. Category

C corresponds to the required number of boxes of each type after the post-solution adjustment is complete. To perform the adjustment, the following information is required.

One 2 x 3 x 1 box = three 1 x 2 x 1 boxes.

One 2 x 3 x 1 box = six 1 x 1 x 1 boxes.

One 2 x 2 x 2 box = four 1 x 2 x 1 boxes.

One 2 x 2 x 2 box = eight 1 x 1 x 1 boxes.

One 1 x 2 x 1 box = two 1 x 1 x 1 boxes.

The above substitution of boxes can be obtained by using the heuristic DP program, and inputing the length and width of the larger box as the dimensions of a pallet.

A dash (-) in Category C indicates that no post-solution adjustment is necessary. Here post-solution adjustment only refers to box substitution. Box reduction is carried out in Category D. The solution in Category C may not satisfy user-specified box proportions. Manual adjustment is also carried out in Category D, if necessary. The solution obtained from either the heuristic DP programming (Category B) or post-solution adjustment (Category C) guarantees a feasible pallet pattern. The numbers of boxes in Categories B and C can be always reduced and still maintain a feasible pallet pattern. However, adding number of boxes to the solutions in categories B or C may result in an infeasible solution. Under this consideration, the number of boxes must be decreased, one by one, until the desired box proportion is obtained. Category D in Table F-1 presents the final solutions used for the simulation.

Consider the following example. Suppose the heuristic DP program generates pallet pattern #10 with the following number of boxes for each size.

Box size	Desired proportion	Number of boxes	Box proportion of the solution
2 x 2 x 2	0.33	2	0.25
2 x 3 x 1	0.67	6	0.75

This DP solution does not yield desired box proportions. The number of boxes of size 2 x 3 x 1 must be reduced. There are 2 boxes of size 2 x 2 x 2. The desired proportion of size 2 x 3 x 1 (66.7%) is twice of that of size 2 x 2 x 2 (33.3%). Therefore, the adequate number of boxes of size 2 x 3 x 1 should be reduced to 4. This combination gives exactly the desired box proportions. The number of boxes in Category D is determined in this manner for the remaining pallet patterns.

For pallet pattern #3, no rational solution is found using the DP heuristic program. One of the possible solutions is to have three boxes for all sizes 1, 3 and 4. This gives a total occupied pallet space of only 45 cubic inches. It is felt that a better solution might exist. Since the 1 x 1 x 1 box is a unit cube, only two box sizes 2 x 2 x 2 and 2 x 3 x 1 need to be considered. By inputing the original number of boxes in Category A (four for each box size) and these two box sizes to the heuristic DP program, the following new solution is obtained.

Four boxes of size 2 x 2 x 2, and

four boxes of size 2 x 3 x 1.

This combination of boxes occupies 56 cubic inches out of a 64 cubic inch pallet. There is 8 cubic inches remaining on the pallet. As many

as eight 1 x 1 x 1 boxes can be loaded to the pallet. Since the box proportion of these three box sizes must be 33.3%, four boxes of size 1 x 1 x 1 are used for pallet pattern 3.

The symbols W and W/O shown in the "DP Program" column of Table F-1 represent the different dynamic programming procedures used to solve for the solution. "W" involves the use of Rule 4 (see Chapter III) to consider the desired number of boxes and current allocated number of boxes. In contrast, "W/O" skips Rule 4 and intends to maximize the pallet space occupied by boxes. In addition, the values under the "Time" column give the computer run time in seconds to reach a DP heuristic solution. All these pallet patterns were determined using an IBM AT microcomputer.

Table F-1. Number of boxes for a pallet cube[a]

Dist. No.	Cat.	Box type 1	2	3	4	DP Program	Time (seconds)
1	A	0	4	4	4	W	35
	B	0	4	4	4		
	C	--	--	--	--		
	D	0	4	4	4		
2	A	0	4	8	0	W	27
	B	0	0	8	0		
	C	0	4	7	0		
	D	0	4	7	0		
3	A	4	0	4	4	W/O	21
	B	16	0	3	4		
	C	--	--	--	--		
	D	4	0	4	4		
4	A	0	10	5	0	W	15
	B	0	8	6	0		
	C	0	12	5	0		
	D	0	10	5	0		
5	A	4	6	6	0	W/O	29
	B	8	0	7	0		
	C	8	8	5	0		
	D	5	5	5	0		
6	A	0	13	0	6	W/O	27
	B	0	14	0	6		
	C	--	--	--	--		
	D	0	12	0	6		

[a] A: Solution of eq. 3.43
B: Solution of the heuristic DP
C: Post-solution adjustment
D: Final solution used.

Table F-1. continued

Dist. No.	Cat.	Box type 1	2	3	4	DP Program	Time (seconds)
	A	4	0	7	0		
7	B	8	0	7	0	W/O	7
	C	--	--	--	--		
	D	4	0	7	0		
	A	7	7	0	7		
8	B	6	8	0	7	W/O	48
	C	8	7	0	7		
	D	7	7	0	7		
	A	0	4	0	9		
9	B	0	8	0	8	W	36
	C	--	--	--	--		
	D	0	4	0	8		
	A	0	0	3	6		
10	B	0	0	2	6	W	11
	C	--	--	--	--		
	D	0	0	2	4		
	A	12	26	0	0		
11	B	12	26	0	0	W	25
	C	--	--	--	--		
	D	12	25	0	0		
	A	0	0	0	10		
12	B	0	0	0	8	W/O	5
	C	--	--	--	--		
	D	0	0	0	8		
	A	4	0	0	10		
13	B	16	0	0	8	W/O	11
	C	--	--	--	--		
	D	4	0	0	8		

Table F-1. continued

Dist. No.	Cat.	Box type 1	2	3	4	DP Program	Time (seconds)
	A	0	0	8	0		
14	B	0	0	8	0	W/O	3
	C	--	--	--	--		
	D	0	0	8	0		
	A	0	32	0	0		
15	B	0	32	0	0	W/O	10
	C	--	--	--	--		
	D	0	32	0	0		
	A	0	0	5	3		
16	B	0	0	4	3	W	12
	C	--	--	--	--		
	D	0	0	4	2		
	A	32	16	0	0		
17	B	32	16	0	0	W/O	23
	C	--	--	--	--		
	D	32	16	0	0		
	A	64	0	0	0		
18	B	64	0	0	0	W/O	3
	C	--	--	--	--		
	D	64	0	0	0		
	A	16	0	6	0		
19	B	16	0	6	0	W/O	7
	C	--	--	--	--		
	D	12	0	6	0		
	A	16	0	0	8		
20	B	16	0	0	8	W/O	11
	C	--	--	--	--		
	D	16	0	0	8		

XV. APPENDIX G. PALLET

PATTERNS AND PRECEDENCE DIAGRAMS

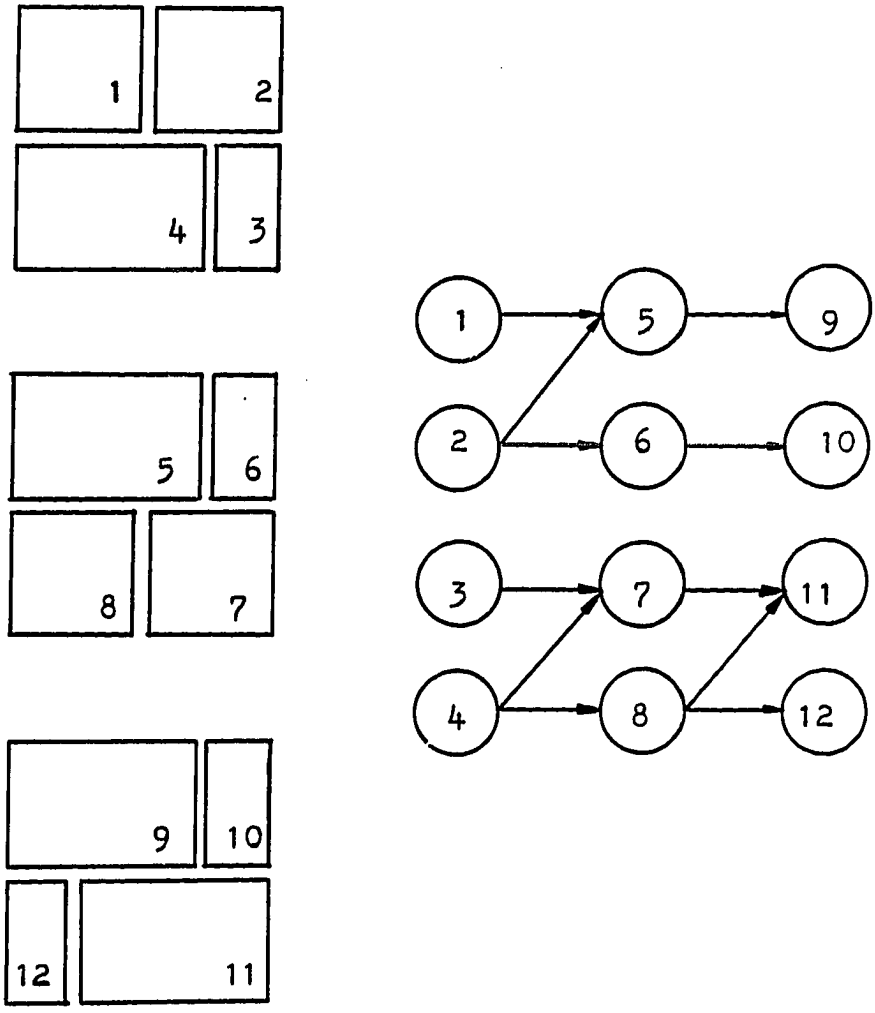


Figure G-1. Pallet pattern 1

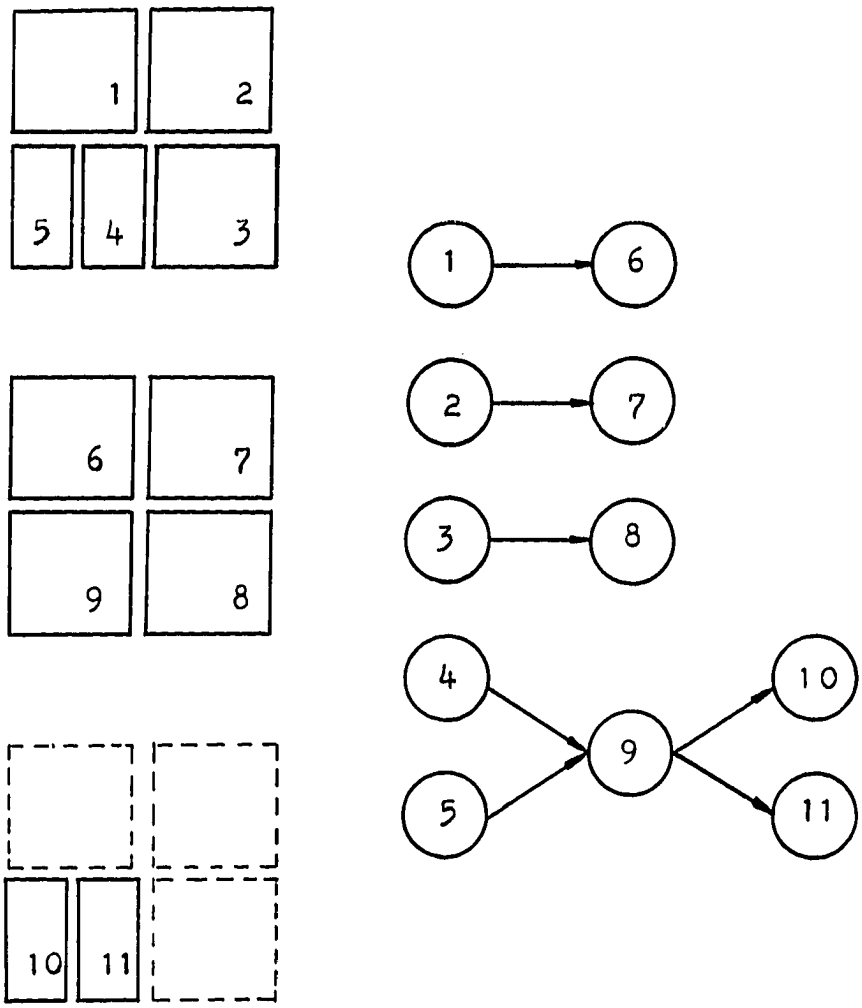


Figure G-2. Pallet pattern 2

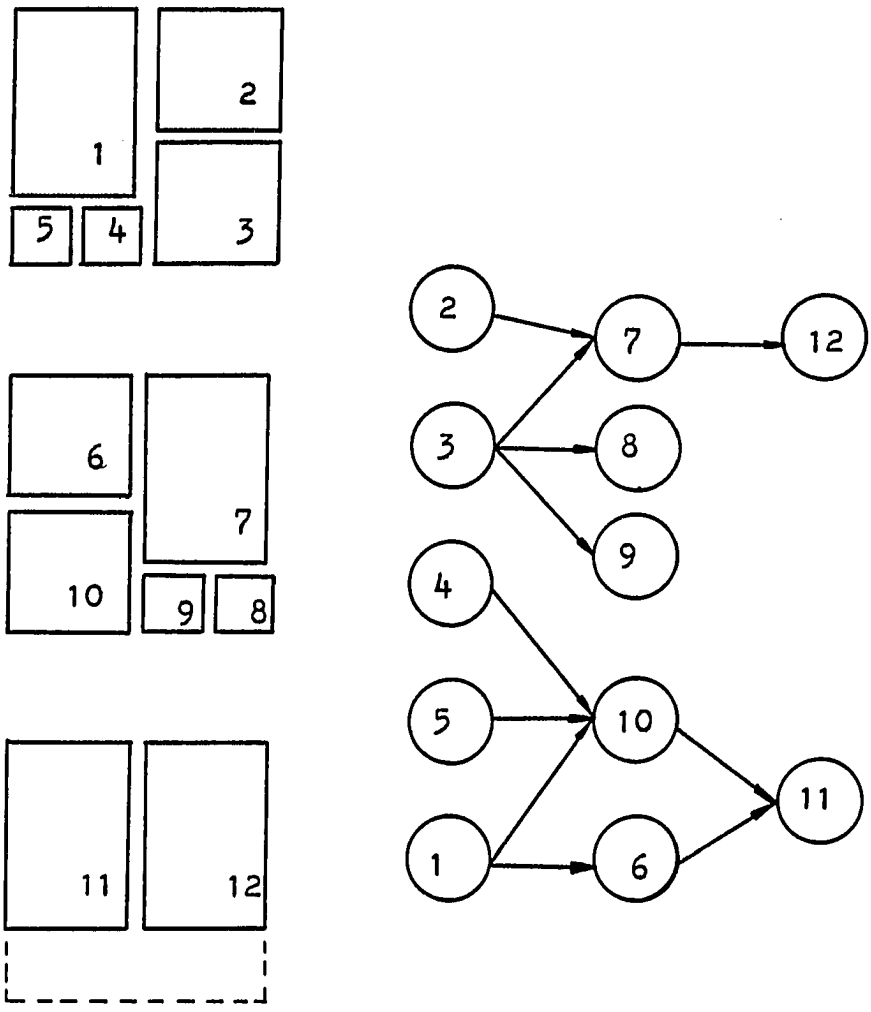


Figure G-3. Pallet pattern 3

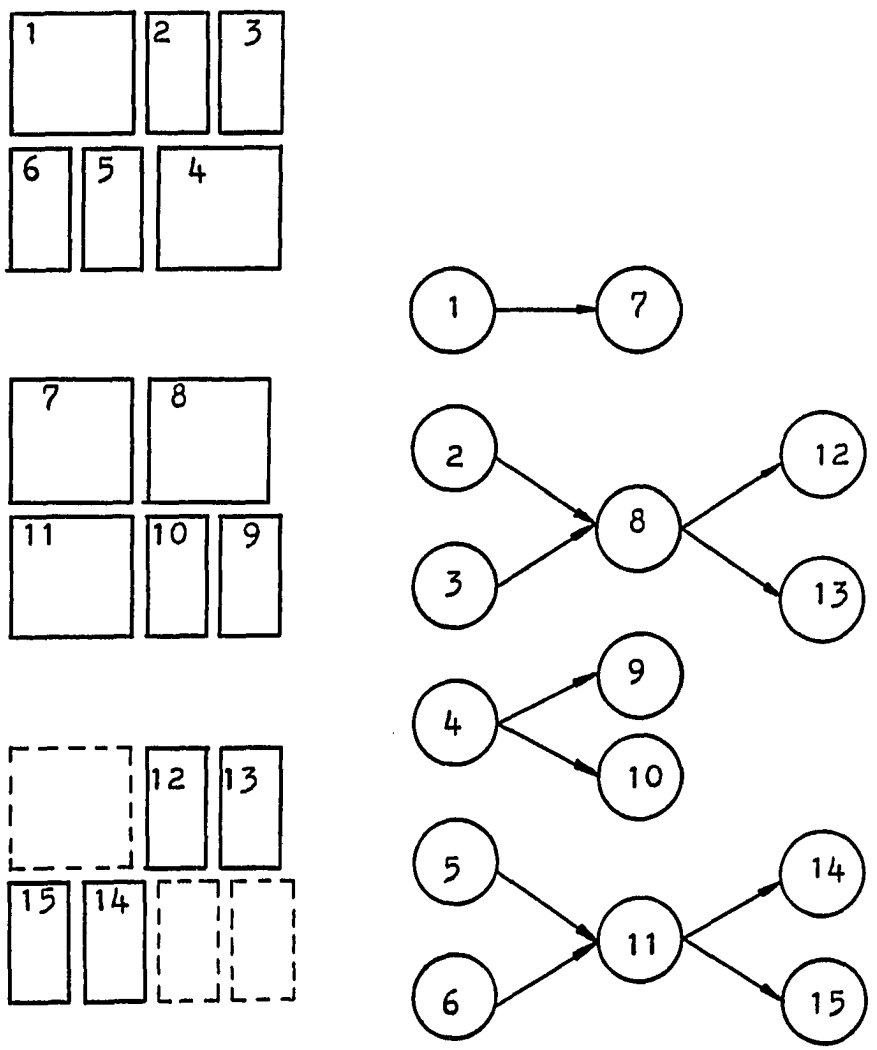


Figure G-4. Pallet pattern 4

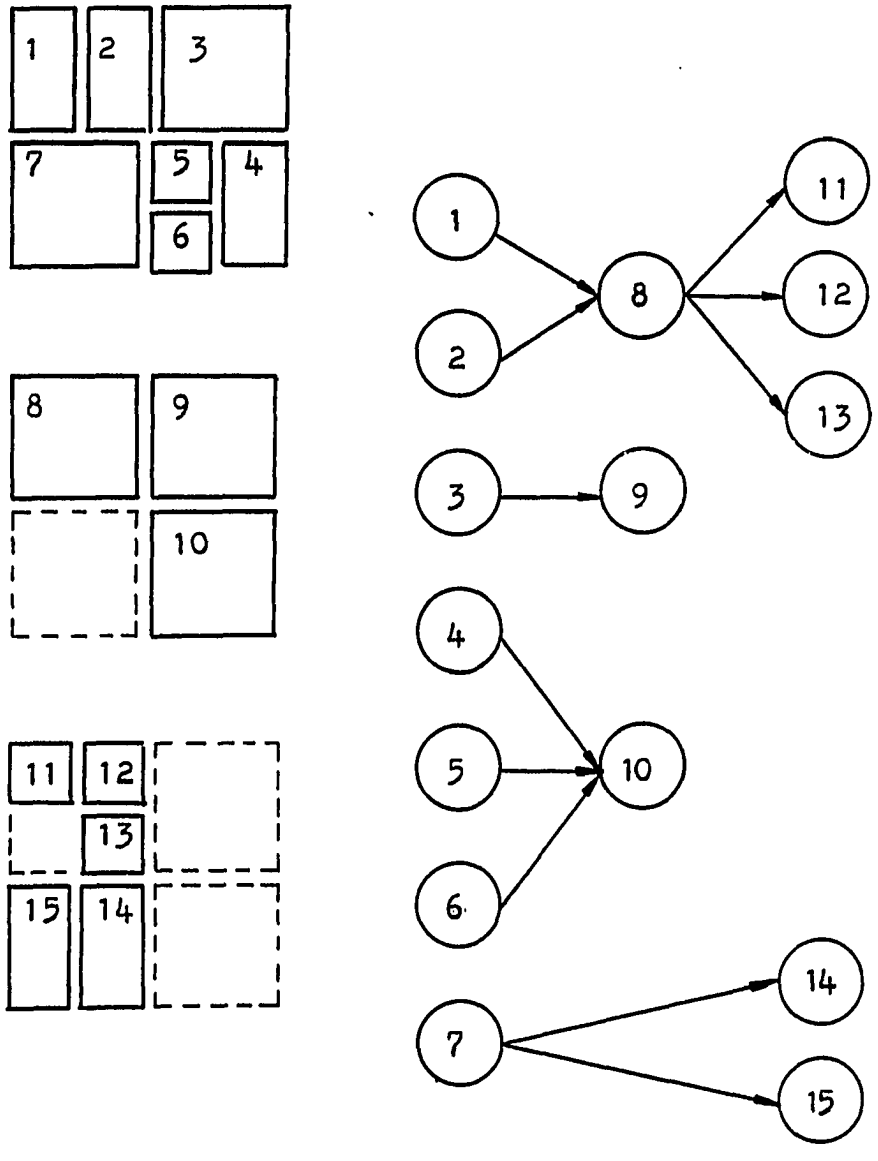


Figure G-5. Pallet pattern 5

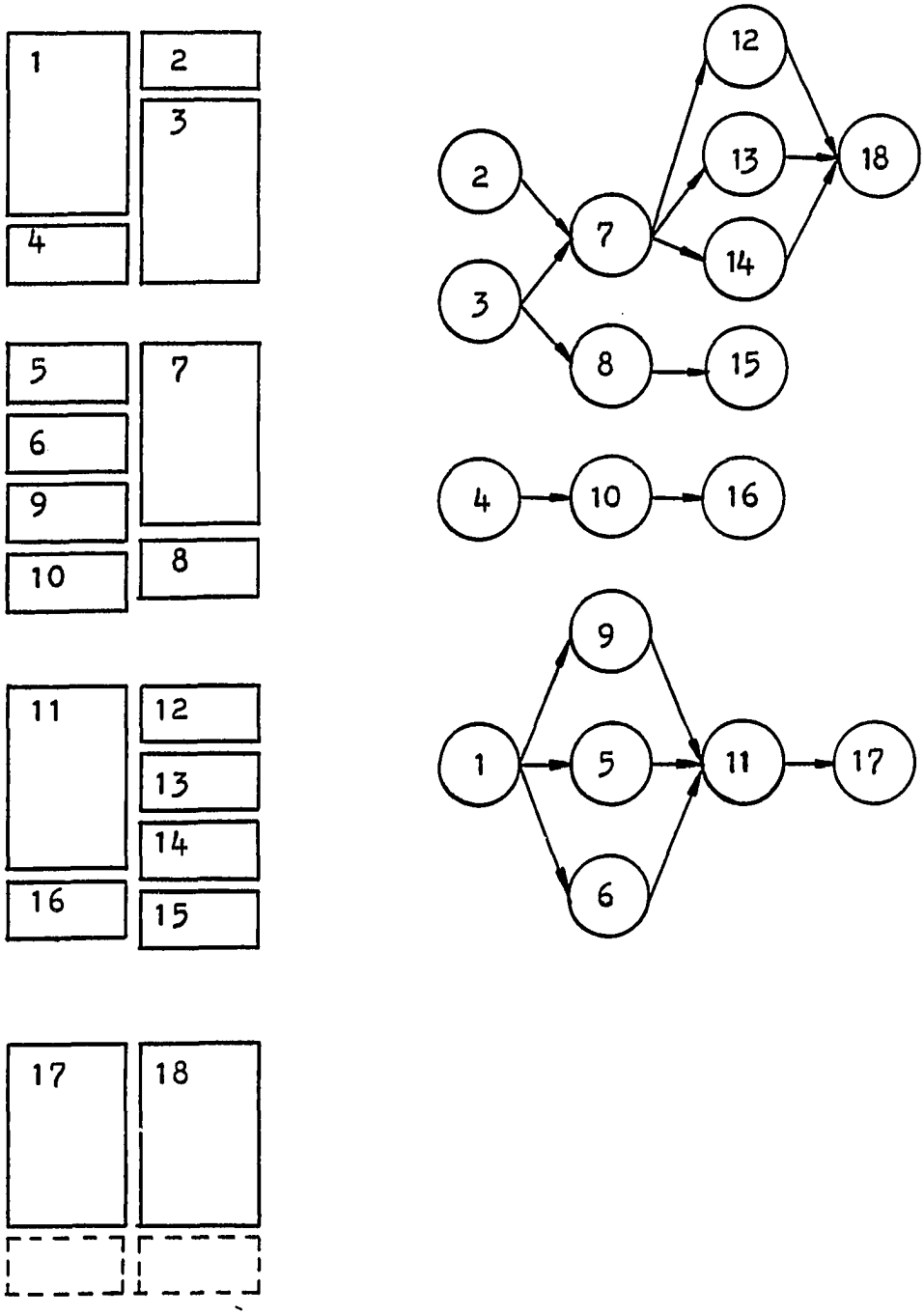


Figure G-6. Pallet pattern 6

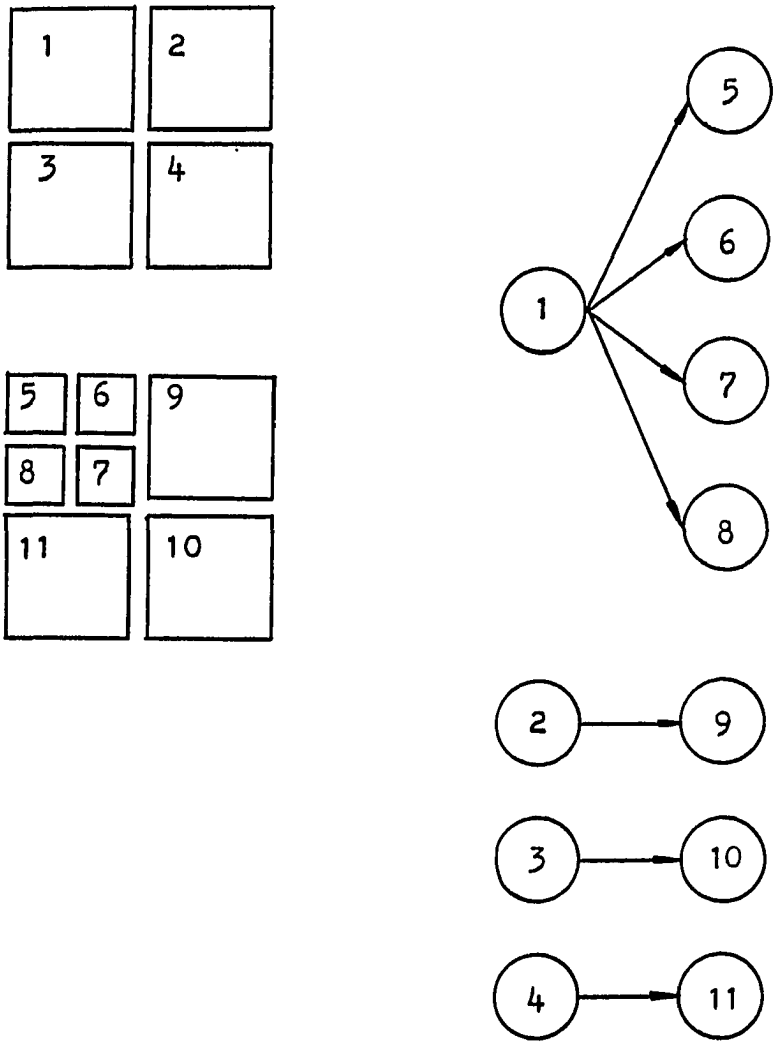


Figure G-7. Pallet pattern 7

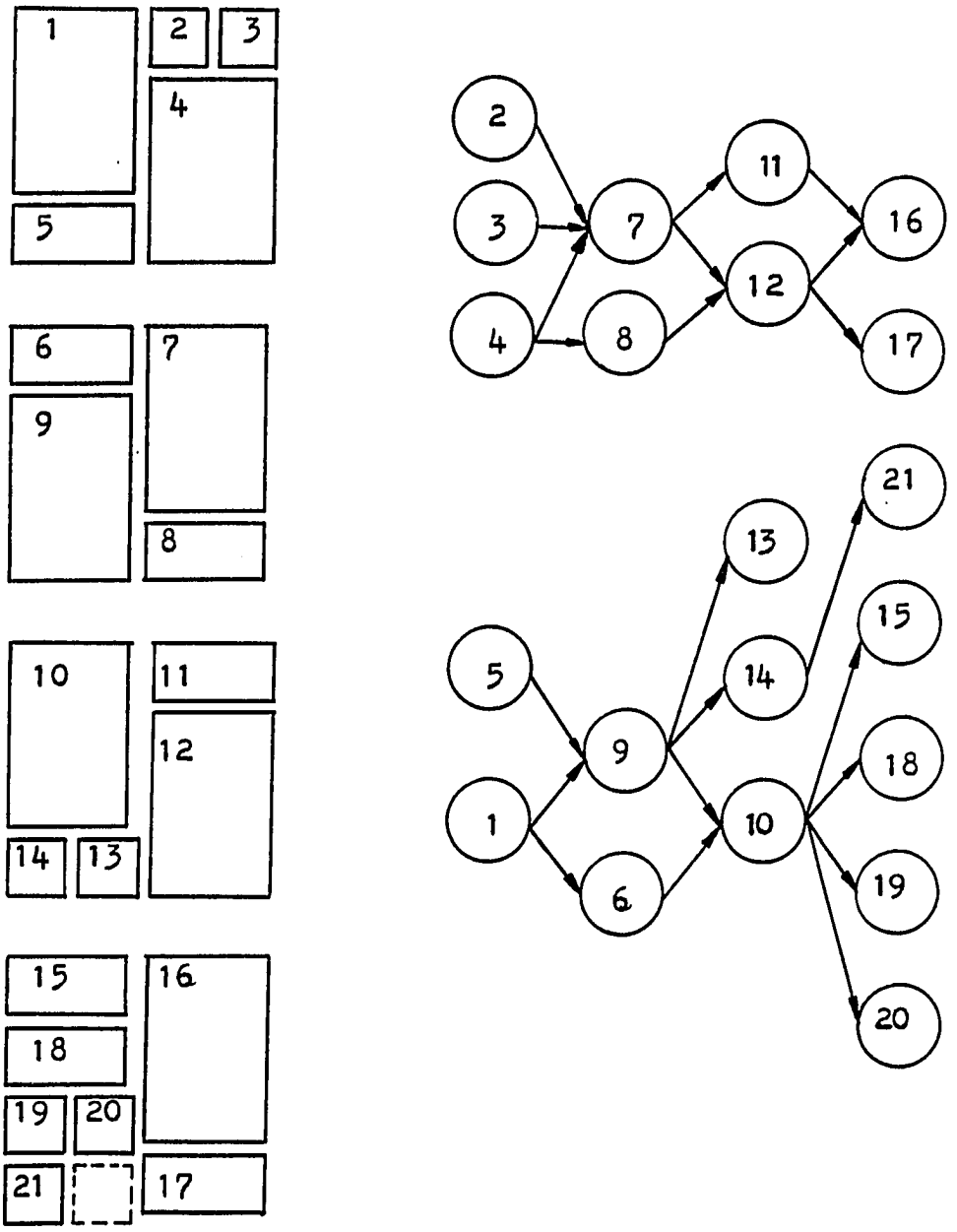


Figure G-8. Pallet pattern 8

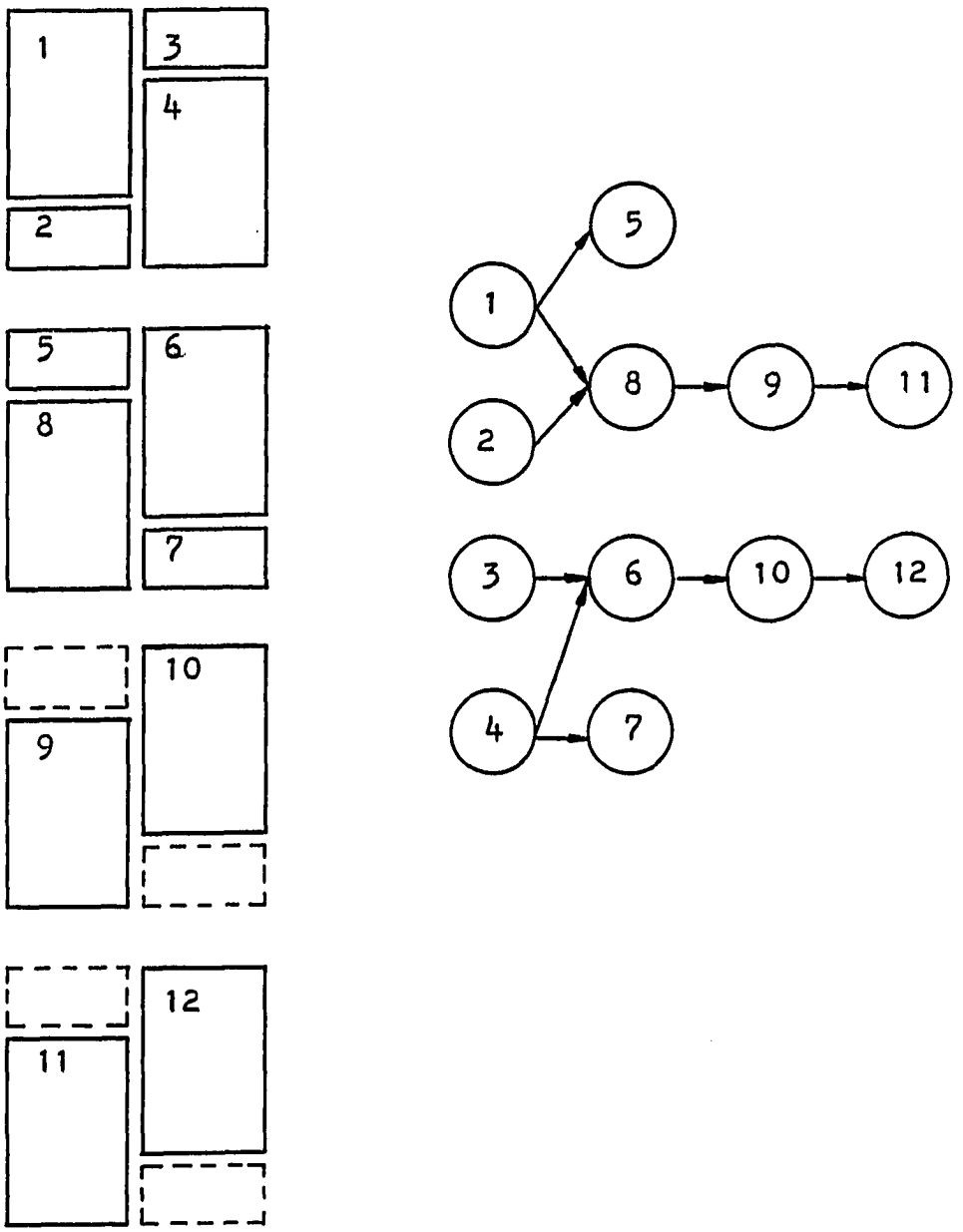


Figure G-9. Pallet pattern 9

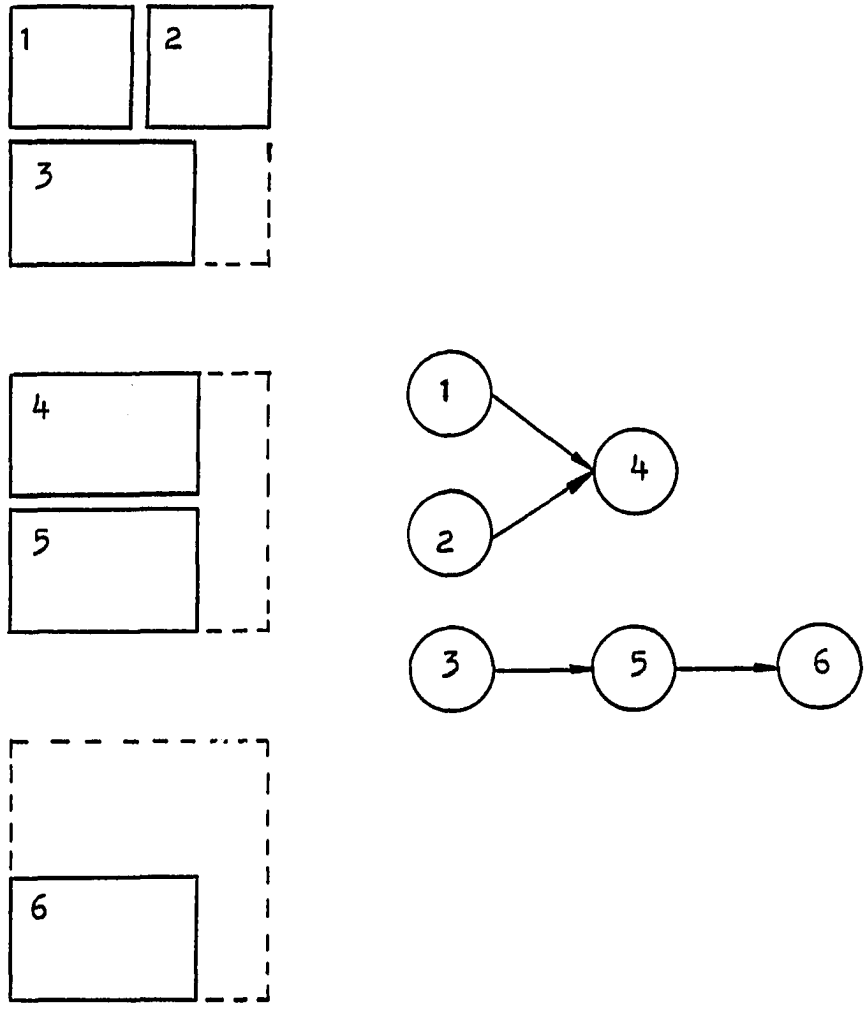


Figure G-10. Pallet pattern 10

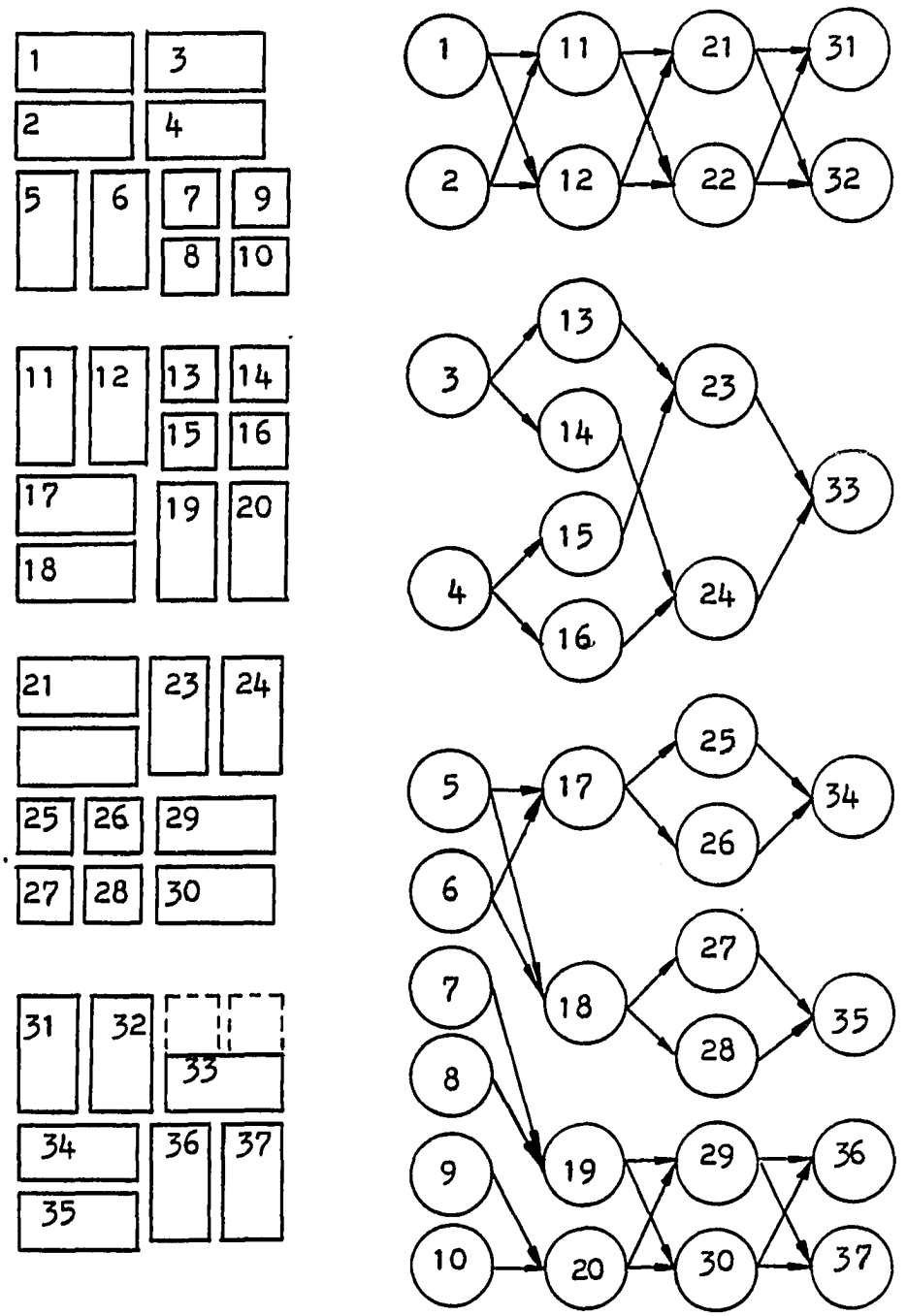


Figure G-11. Pallet pattern 11

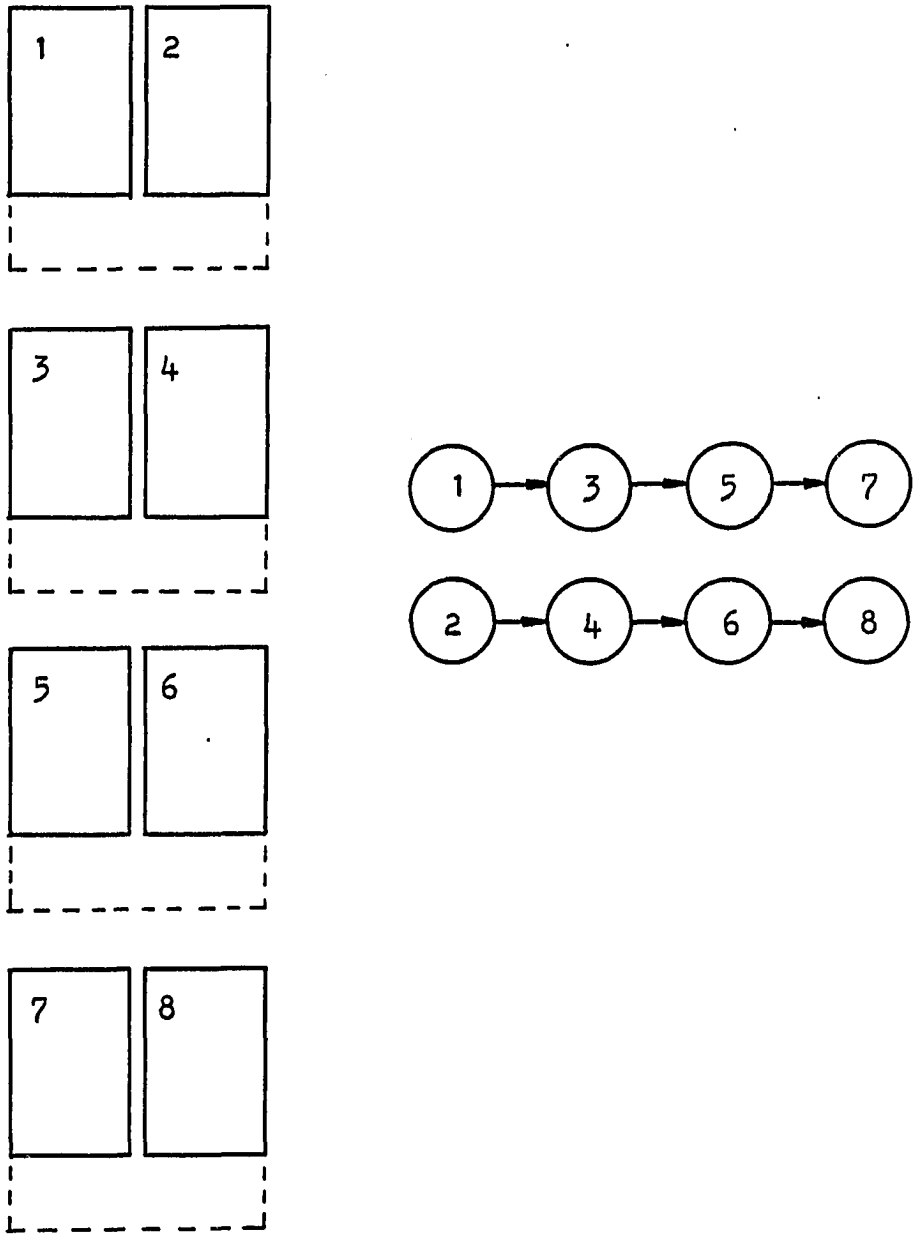


Figure G-12. Pallet pattern 12

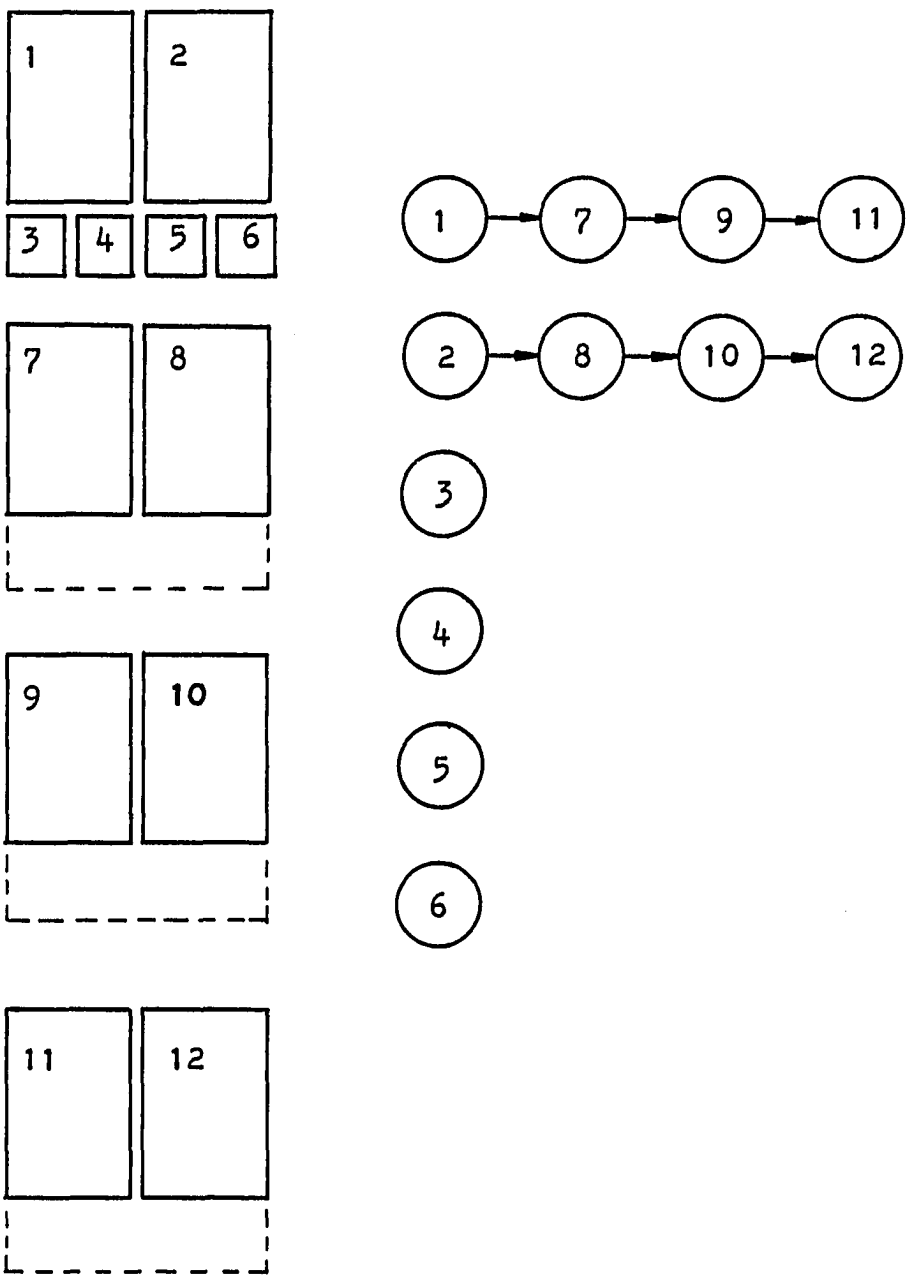


Figure G-13. Pallet pattern 13

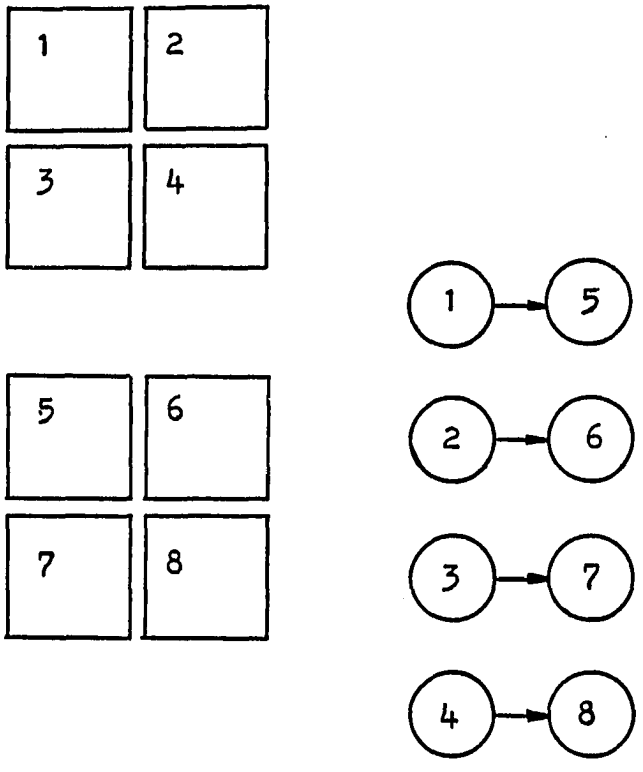


Figure G-14. Pallet pattern 14

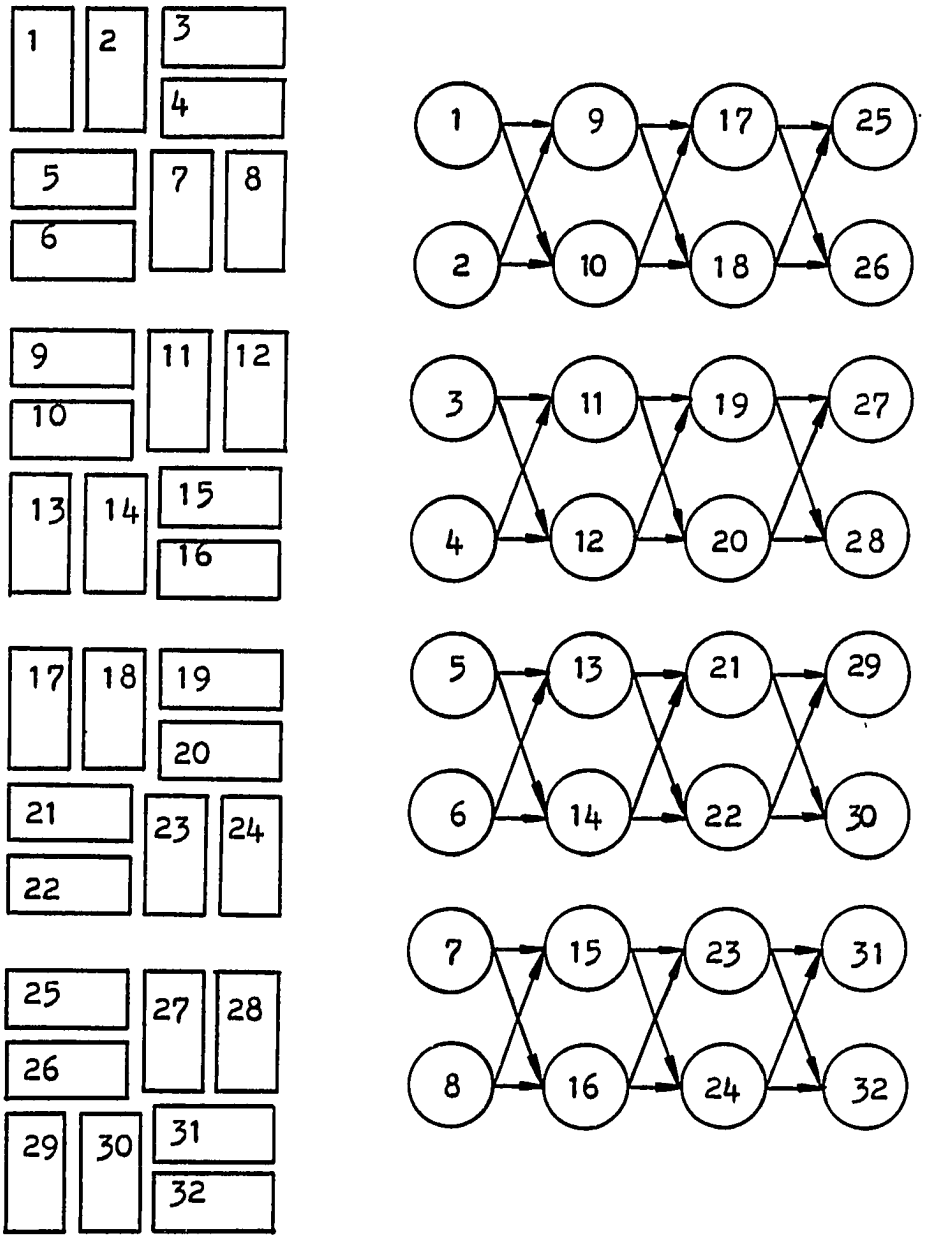


Figure G-15. Pallet pattern 15

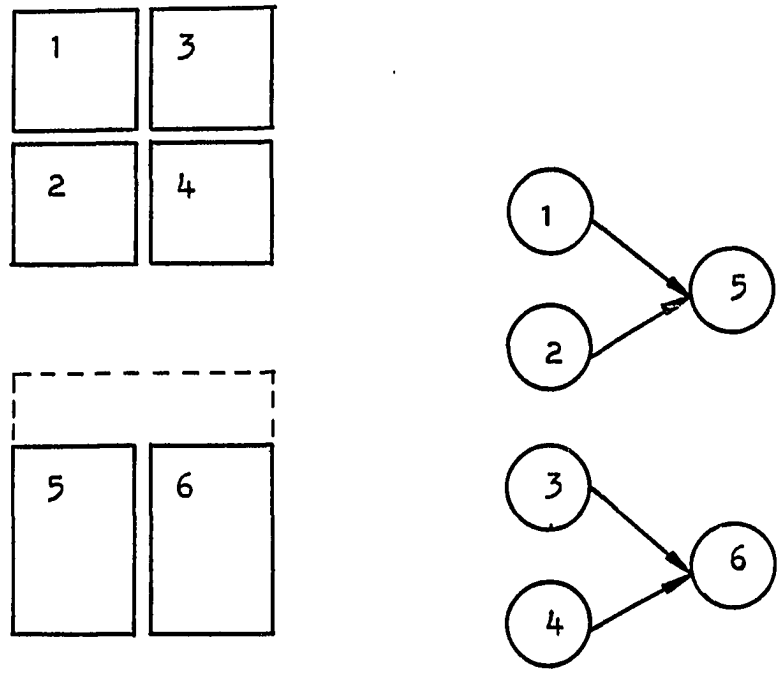


Figure G-16. Pallet pattern 16

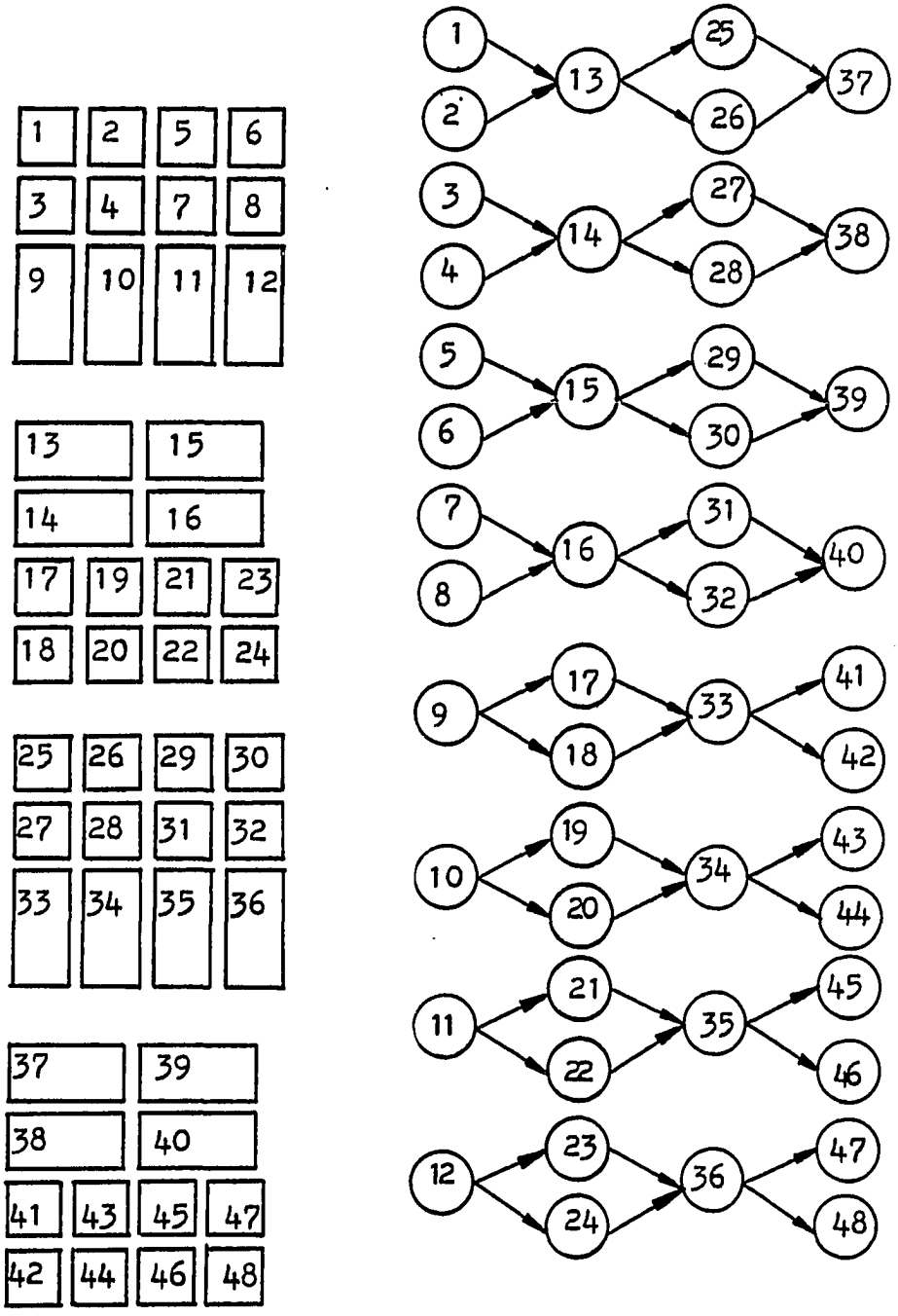


Figure G-17. Pallet pattern 17

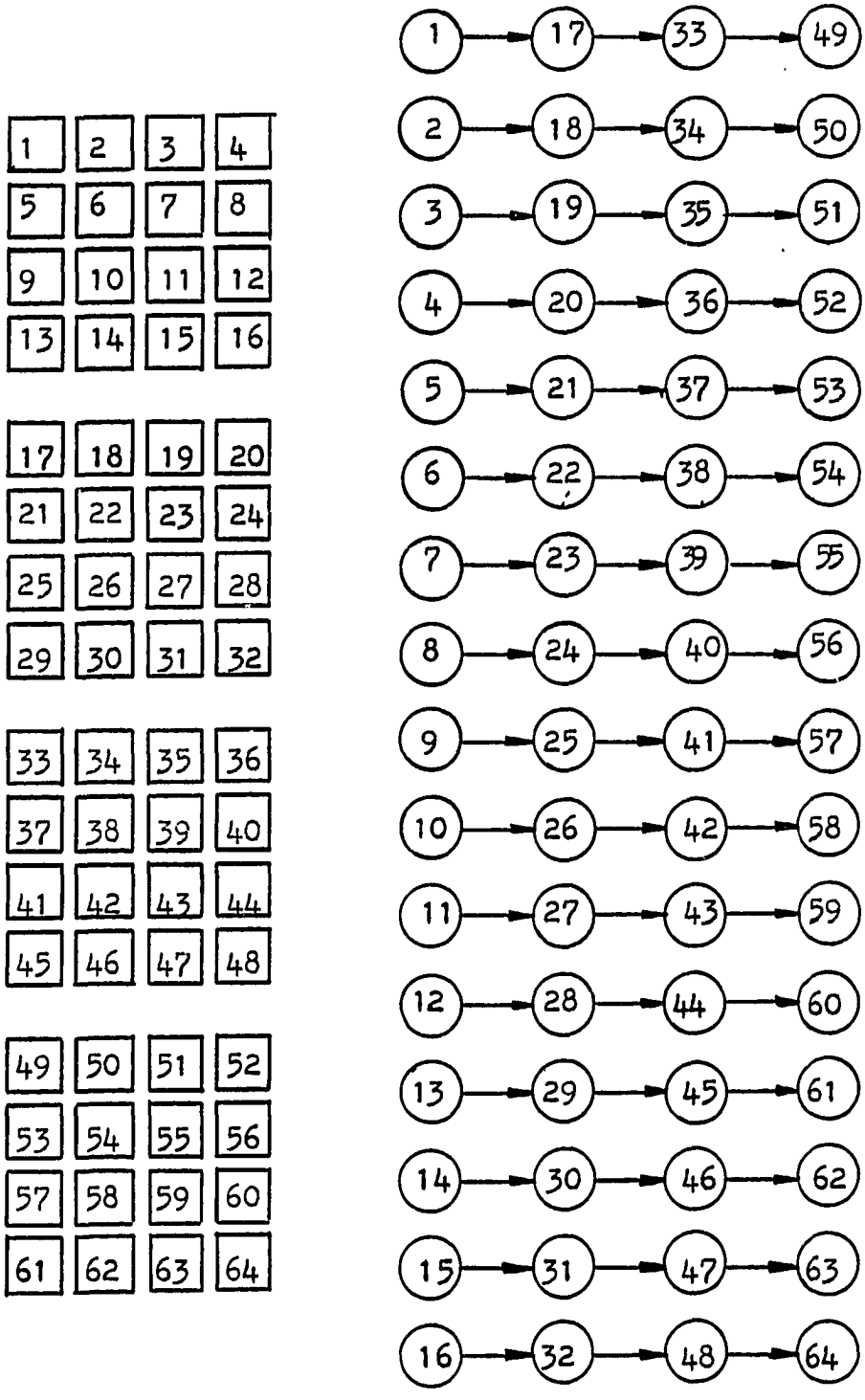


Figure G-18. Pallet pattern 18

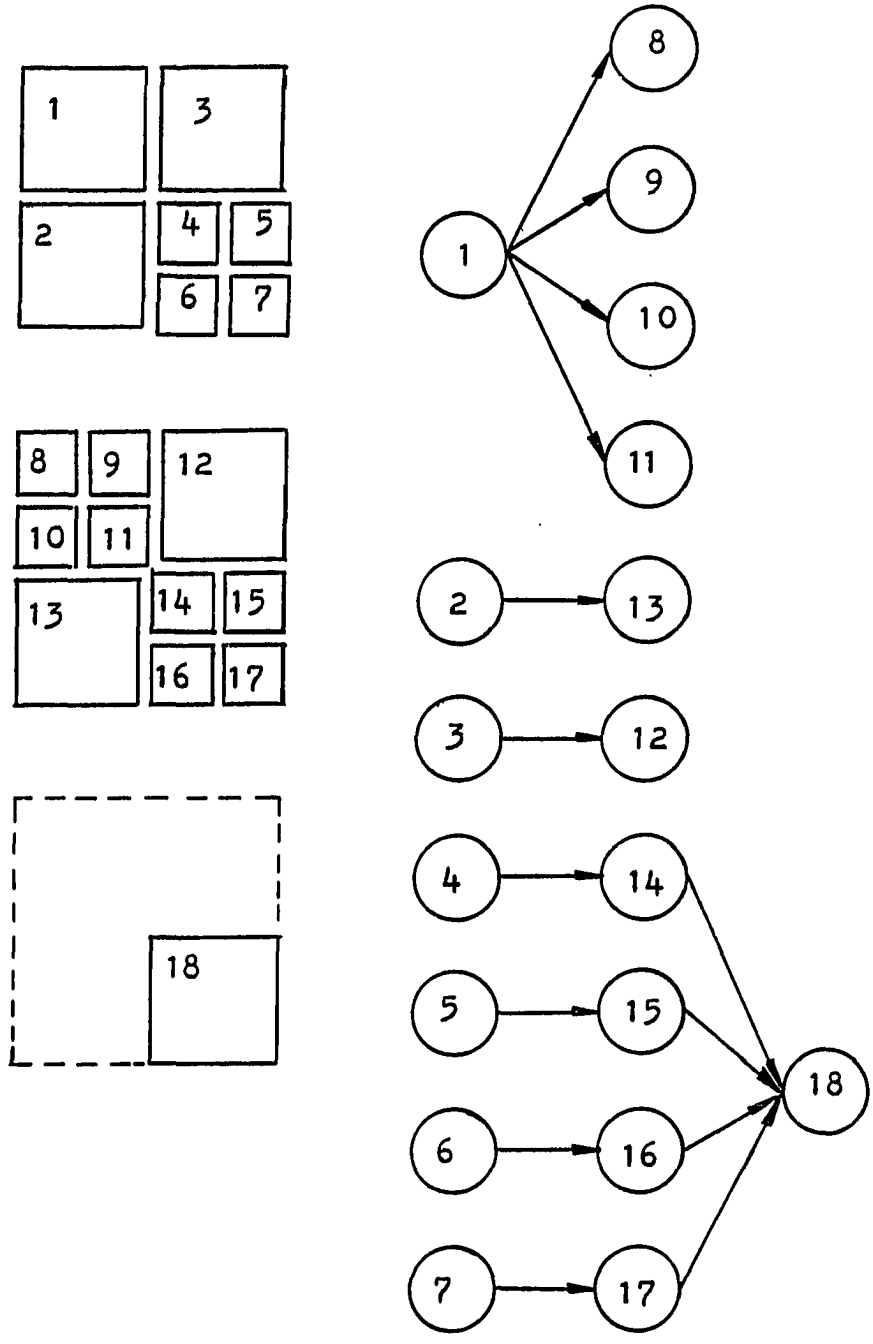


Figure G-19. Pallet pattern 19

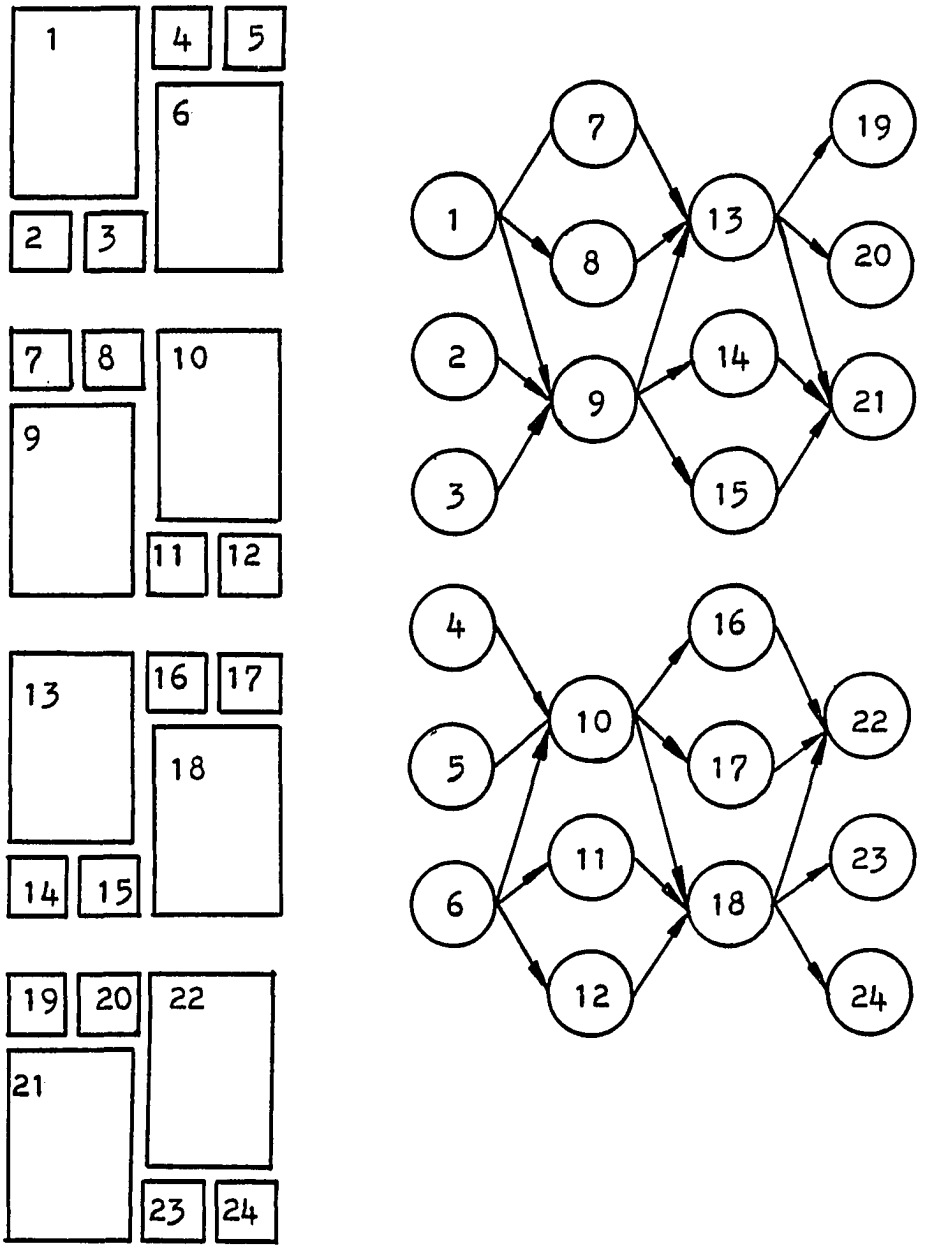


Figure G-20. Pallet pattern 20

XVI. APPENDIX H. VARIATION OF BOXES STORED IN THE STORAGE AREA

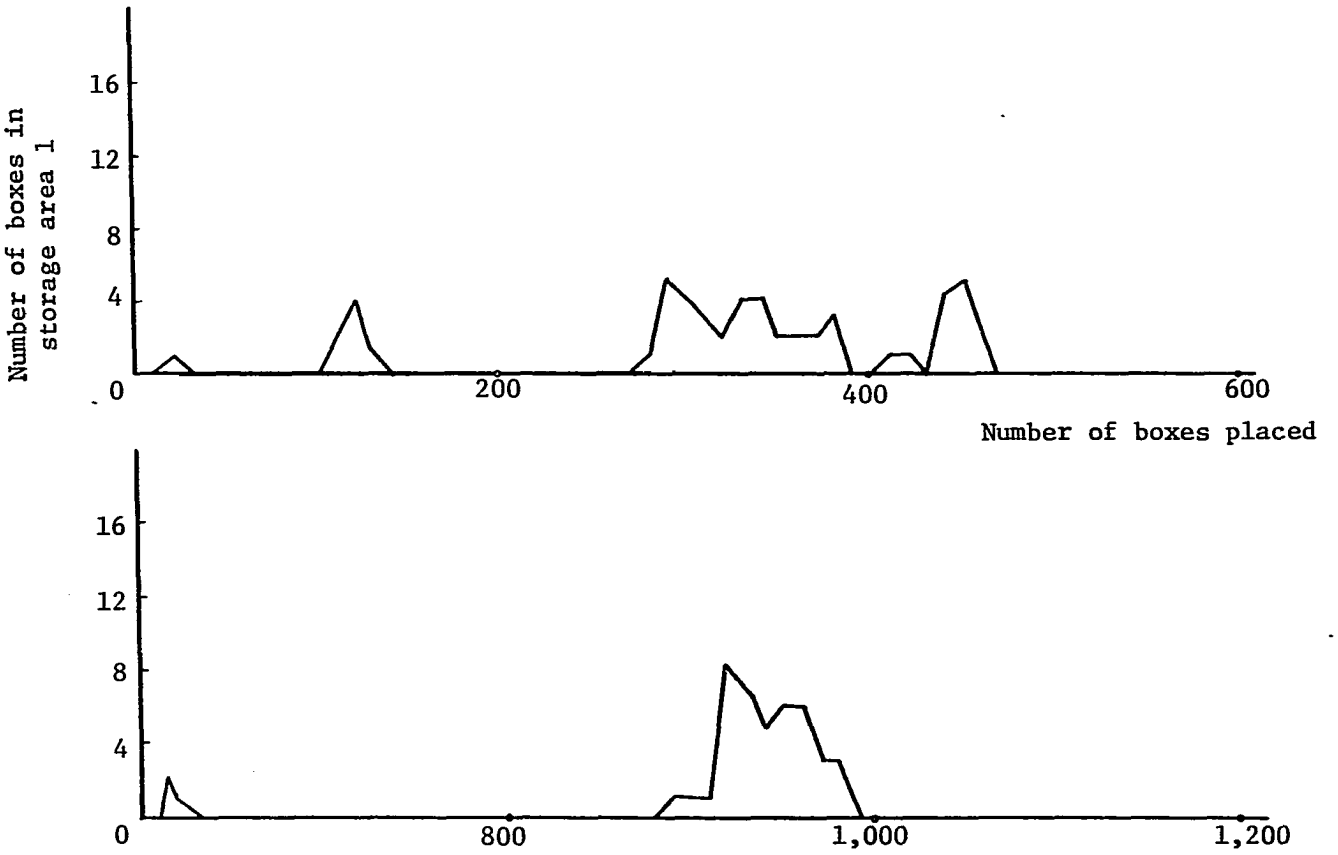


Figure H-1. Variations of stored boxes - box size 1 x 1 x 1, distribution #3

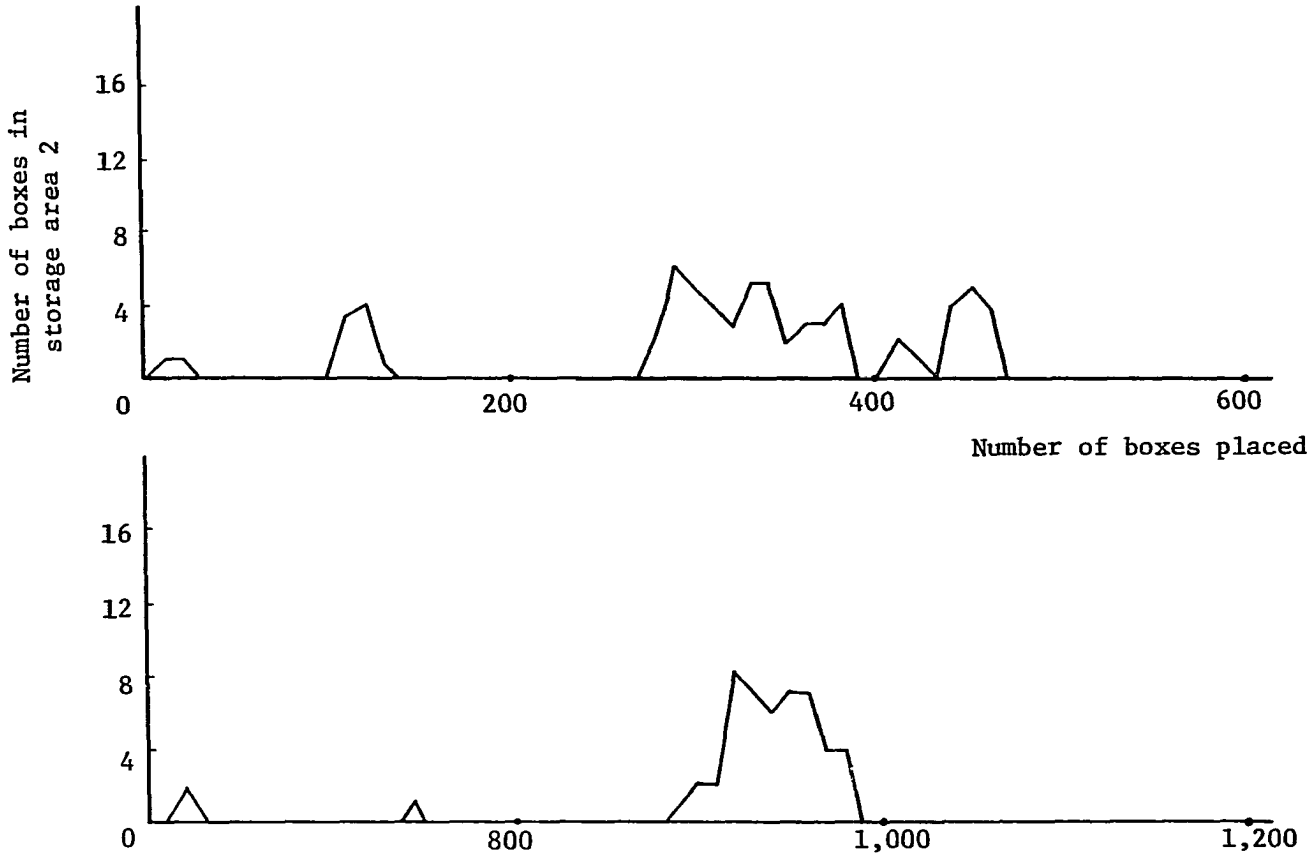


Figure H-2. Variations of stored boxes - box size 1 x 2 x 1, distribution #1

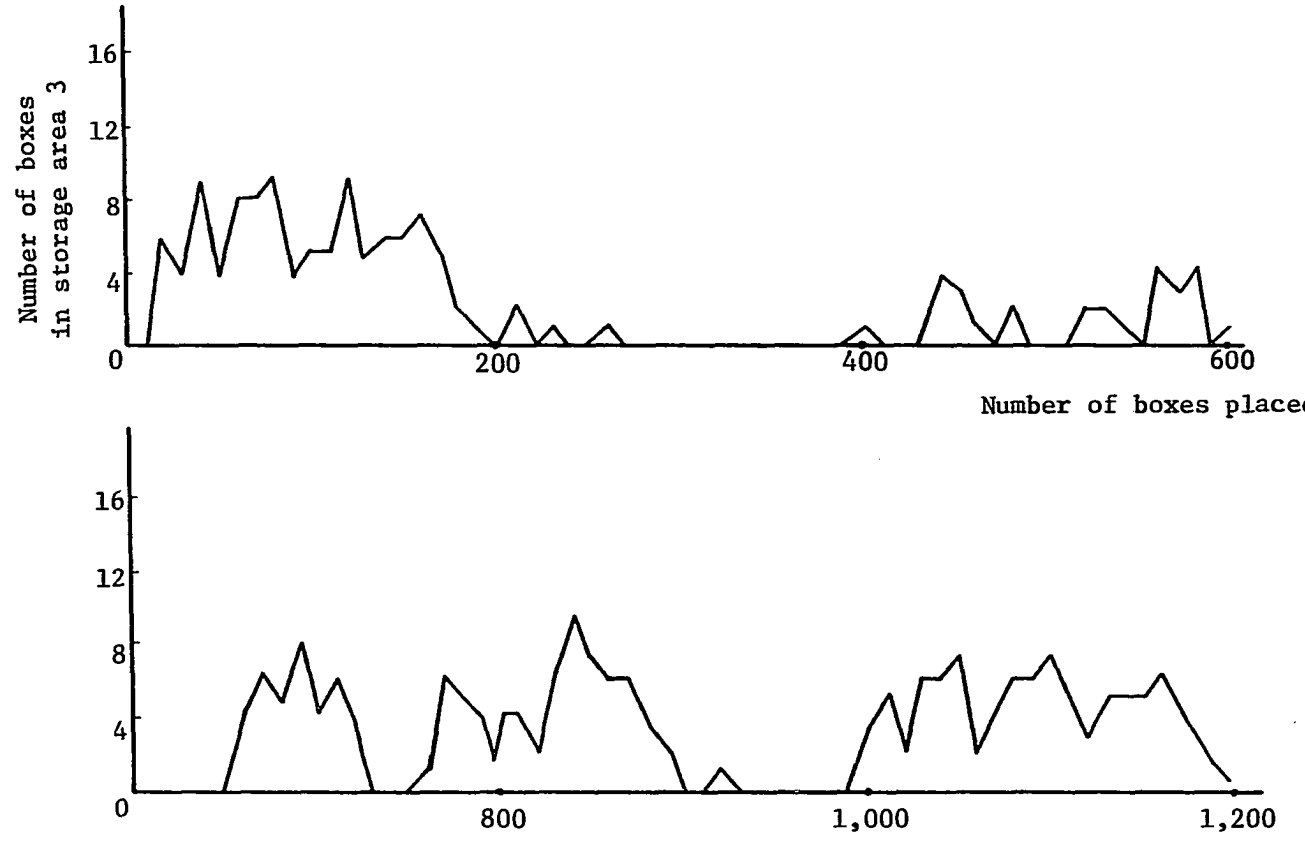


Figure H-3. Variations of stored boxes - box size 2 x 2 x 2, distribution #1

XVII. APPENDIX I.

ROBOT MOVEMENT TIMES

The eleven detailed motions are as follows:

- MOTION 1: move from the hard home position to the location above the pick-up position. Let this location be denoted by ABOVE(PU).
- MOTION 2: move from ABOVE(PU) to the pick-up position. Pick up a box and then move up to ABOVE(PU).
- MOTION 3: move from ABOVE(PU) to the storage area; place the box and then raise the arm to the location above the storage area. Since the locations of storage areas are different, this motion is a function of the box sizes. Let the location above the storage area be denoted by ABOVE(SA).
- MOTION 4a: move from ABOVE(SA) of one box size to the ABOVE(SA) of the desired box size. Pick up a stored box and then raise the arm to the ABOVE(SA) of the desired box size. This motion is a function of the box sizes of which the robot arm moves to and from.
- MOTION 4b: Let the location above the pallet be denoted by ABOVE(PLT). Move the ABOVE(PLT) to ABOVE(SA); lower the arm and pick up a stored box. Raise the arm back to ABOVE(SA). This motion is a function of the box sizes.
- MOTION 5a: move from ABOVE(SA) to ABOVE(PU). This is a function of the box sizes.
- MOTION 5b: move from ABOVE(PLT) to ABOVE(PU).
- MOTION 6a: move from ABOVE(PU) to ABOVE(PLT); lower the arm and place a box. Raise the arm back to ABOVE(PLT). This is a function of the box placement locations on the pallet.

- MOTION 6b: move from ABOVE(SA) to ABOVE(PLT); lower the arm and place a box. Raise the arm back to ABOVE(PLT). This is a function of the box placement locations on the pallet.
- MOTION 7: twist the robot hand 90 degrees.
- MOTION 8: rotate the turntable 90 degrees.

The relationships of these divided motions are shown as a flow in Figure I-1.

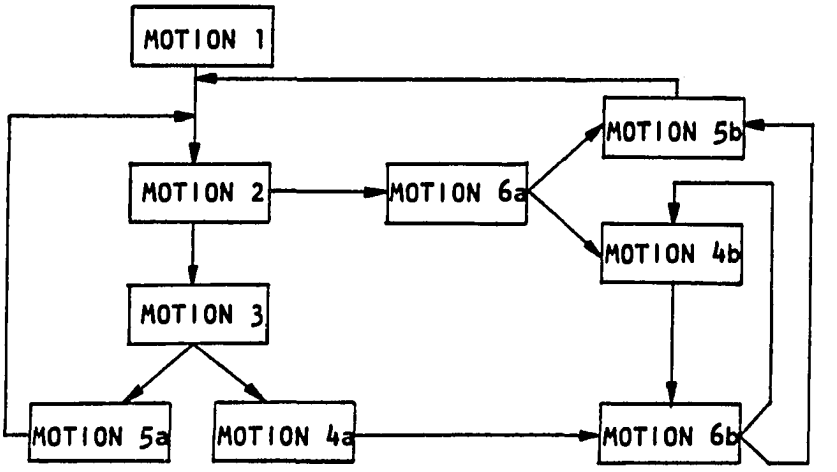


Figure I-1. Relationships of robot motions

With a TI Professional microcomputer, the minimum time unit of the built-in timer is one second. The error of collected movement times is thus in the range of ±1 second. For each robot motion, the mode, the value of the sample that occurs with the greatest frequency, is used as the movement time. The following shows the movement times, all in sec-

onds, for the eleven robot motions.

1) Motion 1: 2 seconds.

2) Motion 2: 3 seconds.

3) Motion 5b: 2 seconds.

4) Motion 7: 3 seconds.

5) Motion 8: 3 seconds per 90 degrees.

6) Motions 3, 4b and 5a

Motion \ Box size	1	2	3	4
Motion 3	6	6	7	9
Motion 4b	7	7	8	9
Motion 5a	2	2	4	5

7) Motion 4a (see Figure 5.2 for the locations of storage areas)

From \ To		Storage area 1	2	3	4
Storage area	1	-	6	6	7
	2	6	-	6	8
	3	6	6	-	5
	4	7	8	5	-

8) Motions 6a and 6b (see the following tables)

Table I-1. Movement times of motion 6a

DIST. NUMBER	MOVEMENT TIMES OF MOTION 6A (ACCORDING TO THE ORDER OF NODE NUMBERS)
1	5 5 5 5 5 5 4 4 4 3 3 4
2	5 4 4 5 5 4 3 3 5 4 4
3	5 5 4 5 5 5 4 3 3 4 4 3
4	5 5 5 5 5 6 4 4 4 4 4 3 3 4 4
5	6 6 4 5 4 5 5 5 4 4 4 4 4 5 4
6	5 6 5 5 5 6 4 5 5 5 5 5 4 4 4 4 4 3
7	5 5 4 5 5 5 4 5 4 3 4
8	5 5 5 4 5 5 5 4 5 5 4 4 4 5 4 4 3 3 4 3 4
9	6 5 6 5 5 5 4 5 4 4 4 3
10	5 4 5 4 5 4
11	6 6 5 4 6 5 5 5 4 5 5 5 5 5 4 5 5 4 4 4 5 5 4 3 5 5 4 5 3 4 4 4 4 4 4 3 3
12	5 4 4 4 5 4 4 3

Table I-1. continued

DIST. NUMBER	MOVEMENT TIMES OF MOTION 6A (ACCORDING TO THE ORDER OF NODE NUMBERS)
13	5 5 6 5 4 4 5 4 4 4 4 3
14	6 5 5 5 4 3 4 3
15	6 5 5 6 5 6 5 4 5 4 4 4 5 5 5 4 4 5 4 4 4 5 4 3 4 4 3 3 4 4 5 3
16	5 5 5 4 5 4
17	5 5 5 5 5 5 5 5 6 5 5 4 5 5 6 4 4 5 5 4 5 4 4 4 5 5 4 4 5 4 4 4 5 5 4 4 5 5 4 4 5 5 4 4 5 4
18	6 6 5 5 6 6 6 5 6 5 5 5 5 5 5 4 5 5 5 4 5 4 4 5 5 5 4 4 6 5 6 6 5 5 4 5 4 4 4 4 4 4 4 3 5 4 3 4 5 4 4 4 4 4 3 3 4 4 3 3 4 4 3 4
19	5 5 4 5 4 5 5 5 4 4 5 4 4 5 4 4 4 3
20	5 6 5 5 5 5 5 5 5 4 5 4 4 5 5 4 5 4 4 4 3 3 4 3

Table I-2. Movement times of motion 6b

DIST. NUMBER	MOVEMENT TIMES OF MOTION 6B (ACCORDING TO THE ORDER OF NODE NUMBERS)
1	6 7 5 8 8 5 7 7 8 5 8 5
2	6 6 7 6 6 6 7 6 6 6 6
3	7 6 7 5 6 6 8 6 6 6 8 8
4	7 6 6 6 6 6 6 7 6 6 7 7 6 6 6
5	6 6 6 6 6 6 6 6 6 7 6 6 7 6 6
6	8 6 8 6 6 7 8 7 7 6 8 7 7 6 6 7 8 9
7	7 6 7 6 6 6 6 6 7 7 6
8	8 6 6 8 6 6 8 7 8 8 5 8 6 6 6 8 7 6 7 6 7
9	8 6 5 8 6 8 6 8 8 8 8 8
10	6 7 7 7 7 7
11	5 6 6 6 6 6 6 6 6 6 6 5 6 6 6 6 6 6 6 6 6 7 6 6 6 6 7 6 6 6 6 6 4 4 4 3 3
12	7 7 7 8 7 8 8 7

Table I-2. continued

DIST. NUMBER	MOVEMENT TIMES OF MOTION 6B (ACCORDING TO THE ORDER OF NODE NUMBERS)
13	8 8 6 7 6 6 8 8 8 8 8 9
14	6 6 6 7 7 7 7 7
15	6 6 6 6 6 6 7 6 6 6 6 6 6 7 7 7 7 6 7 6 6 6 6 6 6 7 6 6 7 6 6 6 7
16	6 7 7 7 8 8
17	7 6 7 6 6 6 7 6 7 7 7 7 6 6 6 6 7 6 6 6 6 6 6 6 6 7 6 6 7 6 6 6 6 7 6 6 7 7 7 7 6 6 7 6 7 6 6 6 6
18	6 6 6 6 7 7 6 6 6 6 6 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 6 6 7 5 4 6 6 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 6 6 6 6 6 7 7 6 6 6 7 6 6 6
19	6 6 6 6 6 6 6 6 6 6 6 6 6 7 5 6 6 6 6
20	8 6 6 6 6 7 6 6 8 8 6 6 8 6 6 6 6 8 6 6 8 8 6 6

XVIII. APPENDIX J. RANDOM SEQUENCE OF 20 BOX SIZE DISTRIBUTIONS

Table J-1. Random sequence of box size distributions

SEQ. RUN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	15	7	5	19	12	4	20	16	13	3	8	18	11	6	14	9	17	10	1	2
2	3	15	8	20	5	10	2	14	12	9	11	13	6	18	7	16	17	4	1	19
3	16	13	11	7	8	5	17	3	1	12	15	14	10	9	4	2	20	18	19	6
4	5	10	6	7	4	13	9	8	3	17	20	1	18	15	14	2	16	19	11	12
5	6	10	20	12	11	5	8	19	16	9	1	18	13	2	7	3	17	4	15	14
6	7	10	16	6	13	12	19	20	11	1	3	17	5	15	18	8	4	9	2	14
7	9	17	2	16	12	5	15	3	11	19	4	8	10	14	20	7	1	18	13	6
8	4	2	13	1	17	9	3	15	16	14	11	20	19	5	18	12	6	7	8	10
9	2	13	16	14	20	10	8	17	19	6	1	9	18	15	4	3	5	7	12	11
10	8	14	13	16	4	19	12	5	3	1	18	20	9	6	17	15	7	10	11	2
11	8	2	7	10	17	5	20	16	3	4	19	12	11	18	6	9	15	13	1	14
12	2	10	1	20	8	3	11	13	12	15	6	5	4	18	9	19	16	17	14	7
13	1	6	12	8	16	17	9	15	2	7	11	10	13	14	19	3	5	4	18	20
14	7	15	20	2	14	5	11	19	4	18	3	8	1	16	12	6	17	13	10	9
15	2	16	13	4	9	18	15	12	7	5	6	17	3	11	8	20	10	19	1	14

Table J-2. continued

SEQ. RUN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
16	15	8	9	1	7	17	20	5	14	18	19	4	11	2	10	16	12	13	3	6
17	5	1	14	10	8	20	15	11	17	3	12	6	18	2	19	4	16	7	9	13
18	15	9	14	8	3	19	11	4	17	16	5	7	6	13	20	2	10	1	18	12
19	7	15	19	18	20	6	10	3	14	16	2	9	11	8	13	4	12	17	1	5
20	5	13	11	9	16	19	3	17	20	12	4	10	14	2	7	15	8	1	18	6
21	2	7	5	14	12	16	17	9	6	11	15	4	10	20	3	1	18	8	19	13
22	3	7	1	18	4	12	9	15	10	19	8	20	2	16	14	5	6	11	17	13
23	8	16	17	10	1	2	7	12	20	3	9	19	13	11	4	15	6	18	14	5
24	13	17	11	6	3	1	15	10	18	14	7	2	8	20	4	9	12	19	16	5
25	14	6	16	19	20	3	15	7	18	10	8	11	4	5	12	9	2	13	17	1
26	1	14	10	11	2	13	4	19	6	16	12	17	9	18	8	20	15	3	7	5
27	11	6	12	16	8	5	17	18	10	13	3	15	4	9	20	2	14	7	19	1
28	12	9	10	19	3	15	5	13	4	20	1	18	14	11	6	17	7	2	8	16
29	11	6	5	13	9	4	2	10	18	7	8	20	3	1	15	16	12	19	14	17
30	7	11	8	5	15	17	4	13	10	3	9	2	14	16	6	20	19	1	18	12

XIX. APPENDIX K. PALLETIZING

STATISTICS OF DISTRIBUTION RUNS

The symbols below have been used in the following listings.

<1> = Total boxes loaded

<2> = Total palletizing time

<3> = Average cycle time (seconds)

<4> = Total operation time in storage (seconds)

<A> = Average contents

<B> = Average waiting time (seconds)

<C> = Total boxes generated

<D> = Maximum contents

<E> = Total entries

<F> = Zero entries

<G> = Current contents

The palletizing statistics of a distribution run in the following listings were collected independently of the previously distributions. All statistics were reset to zero at the beginning of every distribution run.

Dist. number	Statistics (single-pallet)			
	<1>	<2>	<3>	<4>
1	199	5774	28.9	3356
18	200	3398	17.0	3
4	200	4230	21.1	1359
13	201	4045	20.2	1055
2	185	4686	23.4	2693
3	200	5076	25.4	2693
15	200	4009	20.0	2
19	199	4320	21.6	1582
9	197	5616	28.1	2744
7	186	4833	24.1	2557
6	201	5416	27.1	1809
5	203	5137	25.7	2887
12	202	3531	17.6	46
17	200	3767	18.8	335
16	204	5210	26.0	2850
11	200	4277	21.4	962
10	201	5191	26.0	1462
8	200	5065	25.3	2485
14	222	3956	19.8	492
20	200	4104	20.5	1245

Dist. number	Statistics (Single-pallet, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	0
18	0	0	200	0	0	200	0
4	0	0	0	0	0	0	0
13	1.9	114.3	66	8	31	35	0
2	0	0	0	0	0	0	0
3	0.8	61.3	66	10	15	51	0
15	0	0	0	0	0	0	0
19	0	0	133	0	0	133	0
9	0	0	0	0	0	0	0
7	0.04	2.9	66	1	5	61	0
6	0	0	0	0	0	0	0
5	1.9	149.0	66	12	24	42	0
12	0	0	0	0	0	0	0
17	0.1	4.0	133	3	4	129	0
16	0	0	0	0	0	0	0
11	0.3	22.9	66	4	9	57	0
10	0	0	0	0	0	0	0
8	0.7	57.2	66	7	28	38	0
14	0	0	0	0	0	0	0
20	2.6	81.6	133	12	39	94	0

Dist. number	Statistics (Single-pallet, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.3	23.8	66	5	11	55	0
18	0	0	0	0	0	0	0
4	0.1	3.8	133	3	6	127	0
13	0	0	0	0	0	0	0
2	0.04	2.9	66	2	5	61	0
3	0	0	0	0	0	0	0
15	0	0	200	0	0	200	0
19	0	0	0	0	0	0	0
9	0	0	66	0	0	66	0
7	0	0	0	0	0	0	0
6	0	0	133	0	0	133	0
5	0.3	21.5	67	4	11	56	0
12	0	0	0	0	0	0	0
17	0.2	10.2	67	3	7	60	0
16	0	0	0	0	0	0	0
11	0.8	26.2	134	9	20	114	0
10	0	0	0	0	0	0	0
8	1.3	97.0	67	10	36	31	0
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Single-pallet, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	2.8	245.6	67	10	47	20	0
18	0	0	0	0	0	0	0
4	1.7	105.1	67	6	37	30	0
13	0	0	0	0	0	0	0
2	8.0	291.2	134	19	78	56	15
3	2.4	183.6	67	9	51	16	4
15	0	0	0	0	0	0	0
19	6.0	389.2	67	12	48	19	11
9	0	0	0	0	0	0	0
7	14.4	521.1	134	38	84	50	30
6	0	0	0	0	0	0	0
5	2.3	178.2	67	13	55	12	4
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	5.9	233.0	133	19	70	63	5
11	0	0	0	0	0	0	0
10	3.6	286.2	66	10	45	21	9
8	0	0	0	0	0	0	0
14	0	0	200	0	0	200	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Single-pallet, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	1.4	118.6	67	9	38	29	1
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	0.04	1.3	134	2	2	132	0
2	0	0	0	0	0	0	0
3	0.1	10.6	67	3	17	50	0
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	12.7	532.5	134	25	80	54	3
7	0	0	0	0	0	0	0
6	3.3	264.0	67	7	51	14	2
5	0	0	0	0	0	0	0
12	0	0	200	0	0	200	0
17	0	0	0	0	0	0	0
16	8.9	6.9	67	2	14	53	0
11	0	0	0	0	0	0	0
10	0	0	134	0	0	134	0
8	0.3	21.8	67	4	10	57	0
14	0	0	0	0	0	0	0
20	0.01	0.5	67	1	1	66	0

Dist. number	Statistics (double-pallet)			
	<1>	<2>	<3>	<4>
1	200	5245	26.2	2393
18	200	3398	17.0	3
4	200	3608	18.0	289
13	200	3619	18.1	311
2	191	4361	21.8	1667
3	202	4568	22.8	1862
15	200	4010	20.0	2
19	204	3950	19.7	874
9	199	5546	27.7	2615
7	193	4243	21.1	1473
6	201	4635	23.1	619
5	203	4943	24.7	2538
12	200	3481	17.4	0
17	200	3593	18.0	2
16	207	4749	23.7	1975
11	200	3827	19.1	2
10	200	5320	26.2	1804
8	200	4546	22.7	1539
14	200	3480	17.4	0
20	200	3773	18.9	652

Dist. number	Statistics (Double-pallet, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	0
18	0	0	200	0	0	200	0
4	0	0	0	0	0	0	0
13	0.4	24.5	66	4	10	56	0
2	0	0	0	0	0	0	0
3	0.3	18.1	66	6	6	60	0
15	0	0	0	0	0	0	0
19	0	0	133	0	0	R133	0
9	0	0	0	0	0	0	0
7	0.8	50.4	66	6	15	51	0
6	0	0	0	0	0	0	0
5	1.4	108.5	66	10	22	44	0
12	0	0	0	0	0	0	0
17	0	0	133	0	0	133	0
16	0	0	0	0	0	0	0
11	0	0	66	0	0	66	0
10	0	0	0	0	0	0	0
8	0.3	19.9	66	5	11	55	0
14	0	0	0	0	0	0	0
20	1.0	29.4	133	8	21	112	0

Dist. number	Statistics (Double-pallet, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.05	4.2	66	2	3	63	0
18	0	0	0	0	0	0	0
4	0	0	133	0	0	133	0
13	0	0	0	0	0	0	0
2	0	0	66	0	0	66	0
3	0	0	0	0	0	0	0
15	0	0	200	0	0	200	0
19	0	0	0	0	0	0	0
9	0	0	66	0	0	66	0
7	0	0	0	0	0	0	0
6	0	0	133	0	0	133	0
5	0	0	67	0	0	67	0
12	0	0	0	0	0	0	0
17	0	0	67	0	0	67	0
16	0	0	0	0	0	0	0
11	0	0	134	0	0	134	0
10	0	0	0	0	0	0	0
8	1.3	91.3	67	9	30	37	0
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Double-pallet, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	2.0	157.0	67	8	40	27	0
18	0	0	0	0	0	0	0
4	0.2	13.0	67	3	9	58	0
13	0	0	0	0	0	0	0
2	4.1	134.8	134	13	58	76	9
3	2.9	199.8	67	8	51	16	4
15	0	0	0	0	0	0	0
19	0.8	47.8	67	7	23	44	3
9	0	0	0	0	0	0	0
7	3.6	115.3	134	18	36	98	10
6	0	0	0	0	0	0	0
5	2.5	185.7	67	13	56	11	7
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	6.2	222.8	133	18	56	77	0
11	0	0	0	0	0	0	0
10	1.2	94.7	66	5	25	41	0
8	0	0	0	0	0	0	0
14	0	0	200	0	0	200	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Double-pallet, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.7	55.1	67	6	25	42	0
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	0	0	134	0	0	134	0
2	0	0	0	0	0	0	0
3	0	0	67	0	0	67	0
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	11.0	454.5	134	23	75	59	1
7	0	0	0	0	0	0	0
6	0.5	37.9	67	3	17	50	0
5	0	0	0	0	0	0	0
12	0	0	200	0	0	200	0
17	0	0	0	0	0	0	0
16	0.01	0.3	67	1	1	66	0
11	0	0	0	0	0	0	0
10	2.4	96.7	134	14	23	111	0
8	0.1	6.7	67	3	4	63	0
14	0	0	0	0	0	0	0
20	0	0	67	0	0	67	0

Dist. number	Statistics (triple-pallet)			
	<1>	<2>	<3>	<4>
1	200	4700	23.5	1452
18	200	3398	17.0	0
4	200	3437	17.1	0
13	200	3448	17.2	0
2	197	3822	19.1	674
3	203	3757	18.8	478
15	200	4009	20.0	0
19	200	3427	17.1	0
9	200	5478	27.4	2498
7	200	3947	19.7	879
6	200	4220	21.1	0
5	200	3890	19.4	836
12	200	3482	17.4	0
17	200	3593	18.0	0
16	199	4224	21.1	1188
11	200	3825	19.1	0
10	201	4978	24.9	1201
8	200	4234	21.2	933
14	200	3480	17.4	0
20	200	3537	17.7	219

Dist. number	Statistics (Triple-pallet, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	0
18	0	0	200	0	0	200	0
4	0	0	0	0	0	0	0
13	0	0	66	0	0	66	0
2	0	0	0	0	0	0	0
3	0.04	2.5	66	2	2	64	0
15	0	0	0	0	0	0	0
19	0	0	133	0	0	133	0
9	0	0	0	0	0	0	0
7	1.3	80.6	66	8	17	49	0
6	0	0	0	0	0	0	0
5	2.7	159.4	66	11	26	40	0
12	0	0	0	0	0	0	0
17	0	0	133	0	0	133	0
16	0	0	0	0	0	0	0
11	0	0	66	0	0	66	0
10	0	0	0	0	0	0	0
8	0.1	7.5	66	3	5	61	0
14	0	0	0	0	0	0	0
20	0.2	5.5	133	4	7	126	0

Dist. number	Statistics (Triple-pallet, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	66	0	0	66	0
18	0	0	0	0	0	0	0
4	0	0	133	0	0	133	0
13	0	0	0	0	0	0	0
2	0	0	66	0	0	66	0
3	0	0	0	0	0	0	0
15	0	0	200	0	0	200	0
19	0	0	0	0	0	0	0
9	0	0	66	0	0	66	0
7	0	0	0	0	0	0	0
6	0	0	133	0	0	133	0
5	0.01	0.8	67	1	1	66	0
12	0	0	0	0	0	0	0
17	0	0	67	0	0	67	0
16	0	0	0	0	0	0	0
11	0	0	134	0	0	134	0
10	0	0	0	0	0	0	0
8	1.2	74.1	67	8	22	45	0
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Triple-pallet, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	1.2	81.1	67	6	30	37	0
18	0	0	0	0	0	0	0
4	0	0	67	0	0	67	0
13	0	0	0	0	0	0	0
2	0.8	23.7	134	7	23	111	3
3	0.3	18.1	67	4	11	56	0
15	0	0	0	0	0	0	0
19	0	0	67	0	0	67	0
9	0	0	0	0	0	0	0
7	0.7	21.2	134	8	11	123	0
6	0	0	0	0	0	0	0
5	0	0	67	0	0	67	0
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	1.5	47.3	133	9	19	114	0
11	0	0	0	0	0	0	0
10	0.2	12.7	66	3	8	58	0
8	0	0	0	0	0	0	0
14	0	0	200	0	0	200	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Triple-pallet, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.2	16.1	67	3	12	55	0
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	0	0	134	0	0	134	0
2	0	0	0	0	0	0	0
3	0	0	67	0	0	67	0
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	9.2	379.5	134	21	71	63	0
7	0	0	0	0	0	0	0
6	0	0	67	0	0	67	0
5	0	0	0	0	0	0	0
12	0	0	200	0	0	200	0
17	0	0	0	0	0	0	0
16	0.6	36.4	67	4	16	51	0
11	0	0	0	0	0	0	0
10	1.9	72.3	134	12	21	113	0
8	0	0	67	0	0	67	0
14	0	0	0	0	0	0	0
20	0	0	67	0	0	67	0

Dist. number	Statistics (quadruple-pallet)			
	<1>	<2>	<3>	<4>
1	200	4216	21.1	631
18	200	3398	17.0	0
4	200	3437	17.2	0
13	200	3448	17.2	0
2	200	3468	17.3	33
3	200	3471	17.3	42
15	200	4009	20.0	0
19	200	3425	17.1	0
9	200	5382	26.9	2352
7	200	3757	18.8	559
6	200	4220	21.1	0
5	200	3790	18.9	652
12	200	3482	17.4	0
17	200	3593	17.9	0
16	199	4091	20.5	964
11	200	3826	19.1	0
10	201	4776	23.9	848
8	200	4108	20.5	659
14	200	3480	17.4	0
20	200	3419	17.1	0

Dist. number	Statistics (Quadruple-pallet, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	0
18	0	0	200	0	0	200	0
4	0	0	0	0	0	0	0
13	0	0	66	0	0	66	0
2	0	0	0	0	0	0	0
3	0	0	66	0	0	66	0
15	0	0	0	0	0	0	0
19	0	0	133	0	0	133	0
9	0	0	0	0	0	0	0
7	1.4	80.6	66	8	17	49	0
6	0	0	0	0	0	0	0
5	1.7	98.9	66	9	21	45	0
12	0	0	0	0	0	0	0
17	0	0	133	0	0	133	0
16	0	0	0	0	0	0	0
11	0	0	66	0	0	66	0
10	0	0	0	0	0	0	0
8	0.02	1.0	66	1	2	64	0
14	0	0	0	0	0	0	0
20	0	0	133	0	0	133	0

Dist. number	Statistics (Quadruple-pallet, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	66	0	0	66	0
18	0	0	0	0	0	0	0
4	0	0	133	0	0	133	0
13	0	0	0	0	0	0	0
2	0	0	66	0	0	66	0
3	0	0	0	0	0	0	0
15	0	0	200	0	0	200	0
19	0	0	0	0	0	0	0
9	0	0	66	0	0	66	0
7	0	0	0	0	0	0	0
6	0	0	133	0	0	133	0
5	0	0	67	0	0	67	0
12	0	0	0	0	0	0	0
17	0	0	67	0	0	67	0
16	0	0	0	0	0	0	0
11	0	0	134	0	0	134	0
10	0	0	0	0	0	0	0
8	1.0	58.8	67	7	17	50	0
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Quadruple-pallet, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.4	26.0	67	4	12	55	0
18	0	0	0	0	0	0	0
4	0	0	67	0	0	67	0
13	0	0	0	0	0	0	0
2	0.02	0.4	134	1	1	133	0
3	0	0	67	0	0	67	0
15	0	0	0	0	0	0	0
19	0	0	67	0	0	67	0
9	0	0	0	0	0	0	0
7	0.01	0.4	134	1	1	133	0
6	0	0	0	0	0	0	0
5	0	0	67	0	0	67	0
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	0.5	14.5	133	5	12	121	0
11	0	0	0	0	0	0	0
10	0.01	1.0	66	1	1	65	0
8	0	0	0	0	0	0	0
14	0	0	200	0	0	200	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Quadruple-pallet, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.1	6.7	67	2	6	61	0
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	0	0	134	0	0	134	0
2	0	0	0	0	0	0	0
3	0.02	0.8	67	1	1	66	0
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	7.7	308.2	134	19	67	67	0
7	0	0	0	0	0	0	0
6	0	0	67	0	0	67	0
5	0	0	0	0	0	0	0
12	0	0	200	0	0	200	0
17	0	0	0	0	0	0	0
16	0.6	36.4	67	4	16	51	1
11	0	0	0	0	0	0	0
10	1.1	39.4	134	9	18	116	0
8	0	0	67	0	0	67	0
14	0	0	0	0	0	0	0
20	0	0	67	0	0	67	0

Dist. number	Statistics (look-ahead factor=1)			
	<1>	<2>	<3>	<4>
1	160	4985	24.9	2644
18	203	3547	17.7	188
4	215	4758	23.8	2009
13	210	5404	27.0	2845
2	198	4236	21.2	1473
3	200	5357	26.8	2973
15	199	4032	20.1	74
19	161	4081	20.4	1788
9	202	5623	28.1	2719
7	195	4292	21.5	1540
6	209	5495	27.5	2746
5	175	4624	23.1	2628
12	213	3813	19.1	334
17	237	5743	28.7	3308
16	198	4726	23.6	1921
11	194	4590	22.9	1522
10	196	5790	28.9	3116
8	210	6130	30.6	3536
14	200	3501	17.5	0
20	167	4365	21.8	2235

Dist. number	Statistics (Factor = 1, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	0
18	0.03	0.58	200	3	4	196	3
4	0	0	0	0	0	0	0
13	0.01	0.9	66	1	1	65	0
2	0	0	0	0	0	0	0
3	9.6	779.2	66	3	8	58	0
15	0	0	0	0	0	0	0
19	19.6	603.0	133	45	82	51	45
9	0	0	0	0	0	0	0
7	4.5	292.9	66	15	50	16	15
6	0	0	0	0	0	0	0
5	4.5	316.6	66	19	55	11	19
12	0	0	0	0	0	0	0
17	5.3	230.7	133	14	75	58	10
16	0	0	0	0	0	0	0
11	8.3	580	66	19	49	17	19
10	0	0	0	0	0	0	0
8	1.6	145.4	66	8	47	19	7
14	0	0	0	0	0	0	0
20	24.0	788.6	133	46	93	40	41

Dist. number	Statistics (Factor = 1, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.7	55.9	66	7	22	44	2
18	0	0	0	0	0	0	0
4	0.03	1.2	133	2	5	128	1
13	0	0	0	0	0	0	0
2	0.8	50.8	66	4	31	35	3
3	0	0	0	0	0	0	0
15	0.02	0.3	200	3	3	197	3
19	0	0	0	0	0	0	0
9	0.04	3.5	66	2	3	63	0
7	0	0	0	0	0	0	0
6	8.2	340.0	133	20	73	60	2
5	3.5	244.2	67	18	48	19	17
12	0	0	0	0	0	0	0
17	0.04	3.1	67	2	7	60	0
16	0	0	0	0	0	0	0
11	0.02	0.9	134	4	4	130	4
10	0	0	0	0	0	0	0
8	0	0	67	0	0	67	0
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 1, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	4.2	310.9	67	22	39	28	21
18	0	0	0	0	0	0	0
4	2.7	189.6	67	10	48	19	2
13	0	0	0	0	0	0	0
2	0.5	17.2	134	6	18	116	0
3	0.5	39.9	67	5	23	44	2
15	0	0	0	0	0	0	0
19	0	0	67	0	0	67	0
9	0	0	0	0	0	0	0
7	0.01	0.3	134	2	2	132	2
6	0	0	0	0	0	0	0
5	0	0	67	0	0	67	0
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	0.5	19.1	130	4	18	112	2
11	0	0	0	0	0	0	0
10	0.02	2.0	66	1	2	64	0
8	0	0	0	0	0	0	0
14	0	0	200	0	0	200	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 1, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	2.1	155.2	67	17	42	25	17
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	6.0	244.6	134	16	74	60	9
2	0	0	0	0	0	0	0
3	2.1	165.5	67	8	55	12	8
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	3.7	155.1	134	14	73	61	7
7	0	0	0	0	0	0	0
6	0.02	0.2	67	1	1	66	1
5	0	0	0	0	0	0	0
12	0.01	0.09	200	1	1	199	1
17	0	0	0	0	0	0	0
16	0.4	30.0	70	2	36	34	0
11	0	0	0	0	0	0	0
10	3.2	139.4	134	9	77	57	7
8	1.6	144.3	67	9	53	14	4
14	0	0	0	0	0	0	0
20	0	0	67	0	0	67	0

Dist. number	Statistics (look-ahead factor=1.5)			
	<1>	<2>	<3>	<4>
1	188	5710	28.5	3547
18	198	3483	17.4	150
4	211	4628	23.1	1904
13	192	4519	22.6	1742
2	204	4650	23.2	1955
3	185	4830	24.1	2413
15	205	4151	20.7	194
19	200	4881	24.4	2592
9	197	4530	22.6	1414
7	202	4424	22.1	1678
6	203	5132	25.6	1495
5	185	4800	24.0	2796
12	213	3813	19.1	334
17	198	4056	20.3	911
16	202	5206	26.0	2686
11	192	4573	22.9	1521
10	204	5565	27.8	2259
8	201	5277	26.4	2476
14	198	3561	17.8	147
20	203	4870	24.3	2572

Dist. number	Statistics (Factor = 1.5, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	0
18	0.08	1.5	200	6	6	194	6
4	0	0	0	0	0	0	0
13	0.2	5.1	66	4	11	55	4
2	0	0	0	0	0	0	0
3	1.0	25.3	66	10	33	33	10
15	0	0	0	0	0	0	0
19	6.4	157.4	133	16	79	54	14
9	0	0	0	0	0	0	0
7	4.0	266.1	66	15	50	16	15
6	0	0	0	0	0	0	0
5	3.3	242.2	66	14	55	11	7
12	0	0	0	0	0	0	0
17	0.5	15.1	133	13	18	115	13
16	0	0	0	0	0	0	0
11	6.4	446.7	66	19	48	18	19
10	0	0	0	0	0	0	0
8	2.2	175.4	66	9	52	14	9
14	0	0	0	0	0	0	0
20	8.2	300.5	133	19	81	52	12

Dist. number	Statistics (Factor = 1.5, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	4.0	349.4	66	15	53	13	1
18	0	0	0	0	0	0	0
4	1.0	34.4	133	10	32	101	2
13	0	0	0	0	0	0	0
2	0.1	3.4	66	2	9	57	0
3	0	0	0	0	0	0	0
15	0.02	0.3	200	3	3	197	3
19	0	0	0	0	0	0	0
9	1.1	26.0	66	7	30	36	3
7	0	0	0	0	0	0	0
6	0.3	12.8	133	4	13	120	0
5	2.6	184.3	67	13	46	21	12
12	0	0	0	0	0	0	0
17	0.2	13.2	67	5	12	55	5
16	0	0	0	0	0	0	0
11	0.05	1.9	134	6	6	128	6
10	0	0	0	0	0	0	0
8	0.02	1.5	67	1	4	63	0
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 1.5, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.06	5.2	67	4	5	62	4
18	0	0	0	0	0	0	0
4	0.7	16.5	67	5	22	45	1
13	0	0	0	0	0	0	0
2	1.9	44.2	134	7	49	85	0
3	0.1	10.3	67	4	9	58	3
15	0	0	0	0	0	0	0
19	0.02	1.6	67	3	3	64	3
9	0	0	0	0	0	0	0
7	0.08	0.3	134	2	2	132	2
6	0	0	0	0	0	0	0
5	0.02	0.15	67	1	1	66	1
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	0.09	3.6	130	3	11	119	3
11	0	0	0	0	0	0	0
10	0.8	66.4	66	4	24	42	4
8	0	0	0	0	0	0	0
14	0.08	1.5	200	6	6	194	6
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 1.5, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	2.4	206.3	67	13	51	16	7
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	2.8	63.6	134	9	45	89	5
2	0	0	0	0	0	0	0
3	1.5	35.4	67	9	39	28	9
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	0.3	9.5	134	5	13	121	3
7	0	0	0	0	0	0	0
6	1.8	139.7	67	7	28	39	1
5	0	0	0	0	0	0	0
12	0.05	0.1	200	1	1	199	1
17	0	0	0	0	0	0	0
16	1.8	135.9	70	5	60	10	1
11	0	0	0	0	0	0	0
10	2.1	87.8	134	14	32	102	0
8	0.4	28.5	67	4	20	47	0
14	0	0	0	0	0	0	0
20	0	0	67	0	0	67	0

Dist. number	Statistics (look-ahead factor=2)			
	<1>	<2>	<3>	<4>
1	175	5565	27.8	3557
18	206	3603	18.0	248
4	189	4799	24.0	2708
13	190	5037	25.2	2528
2	216	4764	23.8	2173
3	208	5697	28.5	3469
15	173	3925	19.6	516
19	196	4694	23.5	2306
9	185	4566	22.8	1900
7	214	4738	23.7	2027
6	207	r5818	29.1	3376
5	189	4990	24.9	3067
12	228	4185	20.9	742
17	196	4541	22.7	2013
16	201	5206	26.0	2792
11	202	4742	23.7	1843
10	203	6288	31.4	3682
8	177	5375	26.9	3175
14	216	3849	16.1	357
20	201	4790	23.9	2054

Dist. number	Statistics (Factor = 2, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	0
18	0.03	0.5	200	3	4	196	3
4	0	0	0	0	0	0	0
13	0.01	0.6	66	2	2	64	2
2	0	0	0	0	0	0	0
3	0.9	80.2	66	8	30	36	8
15	0	0	0	0	0	0	0
19	4.8	170.4	133	16	72	61	16
9	0	0	0	0	0	0	0
7	3.6	256.6	66	14	51	15	14
6	0	0	0	0	0	0	0
5	3.2	242.3	66	14	55	11	7
12	0	0	0	0	0	0	0
17	0.05	1.6	133	5	5	128	5
16	0	0	0	0	0	0	0
11	0.06	4.5	66	3	5	61	3
10	0	0	0	0	0	0	0
8	0.09	7.0	66	8	8	58	8
14	0	0	0	0	0	0	0
20	0	0	133	0	0	133	0

Dist. number	Statistics (Factor = 2, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	4.6	388.4	66	15	59	7	10
18	0	0	0	0	0	0	0
4	9.8	354.0	133	22	89	44	20
13	0	0	0	0	0	0	0
2	2.5	183.6	66	9	47	19	4
3	0	0	0	0	0	0	0
15	2.0	40.3	200	35	35	165	35
19	0	0	0	0	0	0	0
9	4.0	275.1	66	12	53	13	12
7	0	0	0	0	0	0	0
6	8.8	385.0	133	22	81	52	21
5	3.5	263.9	67	16	52	15	15
12	0	0	0	0	0	0	0
17	2.8	189.8	67	11	60	7	11
16	0	0	0	0	0	0	0
11	5.0	178.9	134	21	51	83	12
10	0	0	0	0	0	0	0
8	3.1	252.5	67	15	47	20	15
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 2, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.03	2.5	67	4	5	62	4
18	0	0	0	0	0	0	0
4	0.05	3.3	67	3	8	59	3
13	0	0	0	0	0	0	0
2	0.05	1.8	134	1	11	123	0
3	0.08	7.1	67	3	8	59	3
15	0	0	0	0	0	0	0
19	0.02	1.2	67	3	3	64	3
9	0	0	0	0	0	0	0
7	0.02	0.9	134	3	3	131	3
6	0	0	0	0	0	0	0
5	0.00	0.1	67	1	1	66	1
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	0.06	2.5	130	5	5	125	5
11	0	0	0	0	0	0	0
10	0.05	5.0	66	2	5	61	2
8	0	0	0	0	0	0	0
14	0	0	200	0	0	200	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 2, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	3.0	248.6	67	15	54	13	11
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	11.6	435.3	134	21	77	57	15
2	0	0	0	0	0	0	0
3	2.9	246.8	67	12	56	11	1
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	0.2	7.9	134	11	12	122	11
7	0	0	0	0	0	0	0
6	0.08	6.7	67	5	11	56	5
5	0	0	0	0	0	0	0
12	0.02	0.4	200	2	3	197	2
17	0	0	0	0	0	0	0
16	3.4	252.3	70	11	70	0	11
11	0	0	0	0	0	0	0
10	6.6	312.1	134	22	80	54	8
8	3.0	245.2	67	19	52	15	19
14	0	0	0	0	0	0	0
20	3.9	277.4	67	9	58	9	5

Dist. number	Statistics (look-ahead factor=2.5)			
	<1>	<2>	<3>	<4>
1	180	5568	27.8	3426
18	148	3497	17.5	1016
4	203	4585	22.9	1836
13	202	4498	22.5	1795
2	193	4726	23.6	2273
3	206	5485	27.4	3405
15	199	4343	21.7	842
19	194	4833	24.2	2598
9	189	4609	23.0	1867
7	230	5046	25.2	2315
6	188	5021	25.1	2000
5	197	5142	25.7	3176
12	232	4284	21.4	841
17	196	5100	25.5	2713
16	191	5028	25.1	2640
11	206	4890	24.4	1801
10	196	5808	29.0	2946
8	195	5911	29.5	3438
14	224	4021	20.1	535
20	203	4870	24.3	2572

Dist. number	Statistics (Factor = 2.5, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	0
18	8.5	149.0	200	65	66	134	0
4	0	0	0	0	0	0	0
13	2.8	190.8	66	11	48	18	10
2	0	0	0	0	0	0	0
3	1.6	133.6	66	12	45	21	12
15	0	0	0	0	0	0	0
19	8.0	289.2	133	20	83	50	20
9	0	0	0	0	0	0	0
7	3.4	259.2	66	15	50	16	15
6	0	0	0	0	0	0	0
5	2.9	228.3	66	13	52	14	9
12	0	0	0	0	0	0	0
17	6.9	266.4	133	24	84	49	24
16	0	0	0	0	0	0	0
11	7.5	556.4	66	21	50	16	21
10	0	0	0	0	0	0	0
8	1.2	106.6	66	10	47	19	10
14	0	0	0	0	0	0	0
20	8.2	300.5	133	19	81	52	12

Dist. number	Statistics (Factor = 2.5, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	3.8	321.9	66	15	57	9	6
18	0	0	0	0	0	0	0
4	0.4	13.2	133	7	14	119	4
13	0	0	0	0	0	0	0
2	0.08	5.9	66	2	8	58	2
3	0	0	0	0	0	0	0
15	1.0	21.2	200	25	28	172	25
19	0	0	0	0	0	0	0
9	3.4	238.1	66	14	52	14	14
7	0	0	0	0	0	0	0
6	3.6	135.3	133	15	49	84	15
5	2.8	215.1	67	15	53	14	15
12	0	0	0	0	0	0	0
17	0.06	4.5	67	6	6	61	6
16	0	0	0	0	0	0	0
11	0.01	0.3	134	2	2	132	2
10	0	0	0	0	0	0	0
8	0.01	0.5	67	2	2	65	2
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 2.5, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.09	7.7	67	7	7	60	7
18	0	0	0	0	0	0	0
4	2.1	146.3	67	11	42	25	7
13	0	0	0	0	0	0	0
2	5.2	183.0	134	15	68	66	13
3	1.0	80.1	67	12	52	15	12
15	0	0	0	0	0	0	0
19	0.02	1.2	67	3	3	64	3
9	0	0	0	0	0	0	0
7	0.01	0.4	134	2	2	132	2
6	0	0	0	0	0	0	0
5	0.00	0.1	67	1	1	66	1
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	0.08	3.3	130	7	7	123	7
11	0	0	0	0	0	0	0
10	0.1	10.7	66	4	7	59	4
8	0	0	0	0	0	0	0
14	0	0	200	0	0	200	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 2.5, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	2.2	186.8	67	13	47	20	7
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	0.1	3.4	134	7	8	126	7
2	0	0	0	0	0	0	0
3	0.04	2.9	67	4	6	61	4
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	0.1	5.2	134	9	9	125	9
7	0	0	0	0	0	0	0
6	0.3	20.6	67	7	17	50	7
5	0	0	0	0	0	0	0
12	0.02	0.4	200	2	3	197	2
17	0	0	0	0	0	0	0
16	3.4	243.5	70	13	70	0	13
11	0	0	0	0	0	0	0
10	5.4	235.2	134	20	65	69	8
8	1.8	158.9	67	8	58	9	8
14	0	0	0	0	0	0	0
20	0	0	67	0	0	67	0

Dist. number	Statistics (look-ahead factor=3)			
	<1>	<2>	<3>	<4>
1	174	5483	27.4	3401
18	156	3620	18.1	1112
4	196	4591	22.9	2135
13	184	4263	21.3	1648
2	224	4926	24.6	2376
3	178	5350	26.7	3399
15	149	3831	19.1	944
19	194	4985	24.9	2778
9	184	4653	23.3	2020
7	230	5097	25.5	2372
6	193	5401	27.0	2942
5	188	5019	25.1	3106
12	243	4555	22.8	1148
17	251	5781	28.9	3357
16	211	5291	26.4	2751
11	205	4978	24.9	2198
10	215	5548	27.7	2006
8	203	r5543	27.7	3232
14	203	3749	18.7	367
20	201	4392	22.0	1474

Dist. number	Statistics (Factor = 3, Storage area 1)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0	0	0	0	0	0	02
18	7.5	135.8	200	62	64	136	62
4	0	0	0	0	0	0	0
13	2.8	184.6	66	16	50	16	16
2	0	0	0	0	0	0	0
3	2.7	219.6	66	17	53	13	17
15	0	0	0	0	0	0	0
19	8.2	305.8	133	24	84	49	24
9	0	0	0	0	0	0	0
7	3.3	256.7	66	16	50	16	16
6	0	0	0	0	0	0	0
5	2.9	221.3	66	13	52	14	10
12	0	0	0	0	0	0	0
17	0.3	12.0	133	15	15	118	15
16	0	0	0	0	0	0	0
11	0.3	24.2	66	9	15	51	9
10	0	0	0	0	0	0	0
8	0.3	24.4	66	5	15	51	5
14	0	0	0	0	0	0	0
20	0.07	2.3	133	2	5	128	0

Dist. number	Statistics (Factor = 3, Storage area 2)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	3.9	324.6	66	15	58	8	10
18	0	0	0	0	0	0	0
4	2.4	84.2	133	12	60	73	11
13	0	0	0	0	0	0	0
2	1.5	110.8	66	7	45	21	5
3	0	0	0	0	0	0	0
15	7.0	134.0	200	64	65	135	64
19	0	0	0	0	0	0	0
9	3.4	236.6	66	15	53	13	15
7	0	0	0	0	0	0	0
6	9.2	375.3	133	26	83	50	26
5	3.0	224.2	67	20	54	13	20
12	0	0	0	0	0	0	0
17	1.9	166.2	67	17	57	10	17
16	0	0	0	0	0	0	0
11	1.8	69.2	134	21	50	84	21
10	0	0	0	0	0	0	0
8	1.7	139.6	67	8	48	19	4
14	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 3, Storage area 3)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	0.1	8.7	67	8	8	59	8
18	0	0	0	0	0	0	0
4	0.3	19.4	67	4	12	55	1
13	0	0	0	0	0	0	0
2	0.1	3.8	134	4	15	119	0
3	0.06	4.5	67	6	6	61	6
15	0	0	0	0	0	0	0
19	0.06	4.2	67	5	5	62	5
9	0	0	0	0	0	0	0
7	0.02	0.9	134	3	3	131	3
6	0	0	0	0	0	0	0
5	0.02	1.9	67	4	4	63	4
12	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
16	0.1	4.4	130	7	7	123	7
11	0	0	0	0	0	0	0
10	4.0	333.5	66	15	50	16	15
8	0	0	0	0	0	0	0
14	0.2	3.4	200	9	9	191	9
20	0	0	0	0	0	0	0

Dist. number	Statistics (Factor = 3, Storage area 4)						
	<A>	<B>	<C>	<D>	<E>	<F>	<G>
1	2.3	189.0	67	13	48	19	8
18	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
13	0.3	10.9	134	13	13	121	13
2	0	0	0	0	0	0	0
3	2.8	229.3	67	15	56	11	15
15	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0
9	0.4	14.4	134	15	16	118	15
7	0	0	0	0	0	0	0
6	0.07	5.4	67	7	7	60	7
5	0	0	0	0	0	0	0
12	0.03	0.8	200	3	5	195	3
17	0	0	0	0	0	0	0
16	1.8	140.0	70	6	61	9	6
11	0	0	0	0	0	0	0
10	0.01	0.2	134	1	1	133	0
8	0.6	53.8	67	8	30	37	8
14	0	0	0	0	0	0	0
20	1.8	116.8	67	6	37	30	0